C Programming for Engineers

A Review of C. Introducing iostream

Nick Urbanik nicku@nicku.org

This document Licensed under GPL—see slide 263

2006 February

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream

ontinuing our tour

ata Types

Expressions

Jiaiemenis

...ayo

3111010

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

auidelines

Who am I? Who are You?	while Statement	
Overview	do statement	
Quick Tour	Avoid Confusing == with =	
Standard Input, Standard Output	Using a constant or single variable as a test condition	
Redirecting Output and Input	while and the null statement	
•	for Statement	
C++ and iostream library	Comparing while and for	
An overview of the iostream library	if and switch Statements	
Continuing our tour	break, continue, goto	
hello.cpp	Functions	
Basic Syntax	Defining Functions	
Input characters without skipping whitespace	Calling Functions	
I/O of other data to standard output and from standard	Using return Value from Functions	
input	Function Parameters	
Reading into strings	Arrays	
Loops	Defining Arrays	
Reading till the end of file	Arrays and Loops	
Infinite Loop while reading	Strings	
Data Types	Pointers	
Integer Types	Pointers as Function Parameters	
Characters	Arrays and Pointers	
Octal, Hexadecimal Output with ostream	Strong relationship between arrays and pointers	
Floating Point Types	Multidimensional Arrays and arrays of pointers	
Named Constants	Arrays of pointers	
Enumerated Types: enum	Memory Allocation	
Expressions	Multidimensional Arrays	
Arithmetic Expressions	Command Line Arguments: argc, argv	
Arithmetic Operators and Precedence	Structures	
All Operators and their Precedence	Passing Structures to Functions	
Relational Expressions	typedef	
Logical Expressions	Reading and Writing Files	
Assignment Expressions	fstream: file input and output	
Increment, Decrement Operators	Error Handling	
Comma Operator	Binary files	
Arithmetic if Expressions	Character I/O	
Bitwise Operators	Reading a Line at a time: getline()	
Casts	I/O of other data to/from Text Files	
Statements	Guidelines	
Simple Statements	Style Guidelines	
Compound Statements	Program Design	

(ロ) (部) (注) (注) 注 り(で)

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

Continuing our

ata Types

Function

Arrave

....

omers

Arrays and Pointers

Multidimensional Arrays and arrays of

tructures

Reading and Writing

Suidelines

Some Things to Read

- ► My name is Nick Urbanik
- ▶ My email address is nicku@nicku.org
- ► The notes for this class are always available at http://nicku.org/c-for-engineers/
- ▶ I taught in Hong Kong for ten years
- ▶ I taught in Macau one year before that
- Before that I wrote software in C++ to collect data for medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

onlinuing our li

ata Types

Function

Arrays

Pointers

Arroya and Bainta

Multidimensional

.. .

tructures

Reading and Writing

auidelines

ome Things to Read

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.ord
- ► The notes for this class are always available at http://nicku.org/c-for-engineers/
- ▶ I taught in Hong Kong for ten years
- ▶ I taught in Macau one year before that
- Before that I wrote software in C++ to collect data for medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

ontinuing our to

ata Types

Lybicasions

Function

Arroug

Dointoro

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

Guidelines

Some Things to Read

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.org
- ► The notes for this class are always available at http://nicku.org/c-for-engineers/
- ▶ I taught in Hong Kong for ten years
- I taught in Macau one year before that
- Before that I wrote software in C++ to collect data fo medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

ontinuing our to

ata Types

_xp103310113

Function

Arrays

Pointers

Arrays and Pointers

Multidimensional

truoturoo

Reading and Writing

Suidelines

ome Things to Read

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.org
- ► The notes for this class are always available at http://nicku.org/c-for-engineers/
- ▶ I taught in Hong Kong for ten years
- ▶ I taught in Macau one year before tha
- Before that I wrote software in C++ to collect data for medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

Continuing our tou

ata Types

- Apr 00010110

Functio

Arrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of

tructures

Reading and Writing

Auidelines

Some Things to Read

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.org
- ► The notes for this class are always available at http://nicku.org/c-for-engineers/
- ▶ I taught in Hong Kong for ten years
- ▶ I taught in Macau one year before that
- Before that I wrote software in C++ to collect data for medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

onunuing our t

ıta Types

-xp103310113

Function

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

auidelines

ome Things to Read

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.org
- ► The notes for this class are always available at http://nicku.org/c-for-engineers/
- I taught in Hong Kong for ten years
- I taught in Macau one year before that
- Before that I wrote software in C++ to collect data for medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our t

ata Types

unction

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

tructures

Reading and Writing

Guidelines

Some Things to Read

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.org
- The notes for this class are always available at http://nicku.org/c-for-engineers/
- I taught in Hong Kong for ten years
- I taught in Macau one year before that
- Before that I wrote software in C++ to collect data for medical experiments

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea: library

ontinuing our to

ata Types

- Apr 00010110

-unction

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

auidelines

Some Things to Rea

- My name is Nick Urbanik
 - Call me Nick
- ▶ My email address is nicku@nicku.org
- ➤ The notes for this class are always available at http://nicku.org/c-for-engineers/
- ▶ I taught in Hong Kong for ten years
- I taught in Macau one year before that
- Before that I wrote software in C++ to collect data for medical experiments

How we will work in these classes

- ► We will learn by doing
 - I will explain something for a short time
 - you will then try it out for a short time
- Next week:
 - I will print the notes out for you, now that I know what you need
 - I will give you more information about this course
- This is always available from http://nicku.org/; click on the link on the left "C for Engineers".
- I will publish all the teaching material also on http://gonzo.org.au/, and I'll explain to you how to use it

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

onunuing our to

ita Types

_xpr00010110

latements

--, -

. . . .

Multidimensional

Multidimensional
Arrays and arrays of
pointers

ructures

eading and Writing

Guidelines

Overview

Quick Tour

All Operators and their Precedence

Arrays and Loops

Engineers Nick Urbanik

C Programming for

Quick Tour



C — Quick tour

C was originally designed as a low-level systems programming language for an early version of the UNIX operating system.

- Combine:
 - efficiency and hardware access capability of assembler with
 - high level language structures and
 - portability
- Most of the Linux and UNIX operating systems are written in C.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

Quick Tour

Output
Redirecting Output and

++ and lostreal brary

Johnnang our t

tatements

ntore

inters

rays and Point

......

ays and array inters

structures

Files

Guidelines

Some Things to F

C is **portable**

- Although C matches the capabilities of many computers, it is independent of any particular machine architecture.
- it is not difficult to write portable programs that can be run without change on a variety of hardware
- the standard makes portability issues explicit, and prescribes a set of constants that characterize the machine on which the program is run.
 - See limits.h

C Programming for Engineers

Nick Urbanik

Quick Tour

C — quick tour (continued)

- The standard library is a suite of ready-written commonly used functions
- header files declare the library functions, and any symbolic constants required.
- ➤ The (obsolete) header file for the iostream library is included into your program file with:

#include < iostream>

- Header files tell the compiler about the library functions and about the type of the parameters and return types
- linker automatically links in the required functions at compile-time.
- ► C programs make use of functions from the standard library; e.g., all input and output operations are usually performed using standard library functions.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

Quick Tour

Standard Input, Standard Output Redirecting Output and

-+ and iostream rary

ata Types

xpressions

.....

.....

inters

Onitors

rrays and Poi

lultidimensional rrays and arrays o

....

structures

Reading and Writin Files

Guidelines

Samuel Thinness An

Overview

Standard Input, Standard Output

All Operators and their Precedence

Arrays and Loops

Engineers Nick Urbanik

C Programming for

Standard Input, Standard

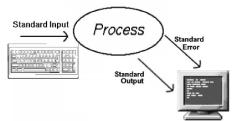
Output

Standard Input, Output

▶ If you run a program in Windows or Linux, it usually has three files already open:

Name	File Descriptor	normally connected to
Standard Input	0	keyboard
Standard output	1	screen
Standard error	2	screen

These are normally connected to the keyboard and your command prompt window



C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

Standard Input, Standard Output

Redirecting Output and input

rary

ata Types

expressions

atements

ravs

intoro

Uniters

Arrays and I

Multidimensiona

truoturoo

Reading and

Guidelines

Some Things to F

Overview

Redirecting Output and Input

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers Nick Urbanik

Redirecting Output and

Input

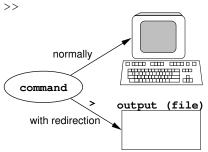
Output Redirection

- ▶ We redirect output using '>'
- For example:

C:\STUDENT> command > output

Creates the file output (or overwrites it if it already exists) and places the standard output from command into it

We can append to a file rather than overwriting it by using



C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

Input

Quick Tour Standard Input

Output
Redirecting Output and

C++ and iostre

Continuing our tour

ata Types

tatements

unctions

iays

ointers

Arrays and I

Multidimensiona Arrays and arra

Structures

Reading and Writi

Guidelines

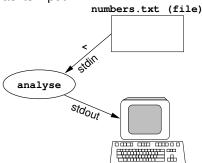
Some Things to

Input Redirection

< redirects standard input from a file, e.g.,</p>

C:\STUDENT> analyse < numbers.txt</pre>

▶ analyse now take the contents of the file numbers.txt as its input



C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

Quick Tour

Standard Input, Standard

Redirecting Output and Input

rary

lata Tunas

Expressions

Statements

unctions

rrays

ointers

Arrays and F

Multidimonoio

rrays and array pinters

Structures

Reading and Writing Files

Guidelines

Some Things to F

C++

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

iostream HDFA

Continuing our tou

.,,,,,,,,

Statements

unctions

. ayo

inters

Arrays and Pointers

ultidimensional rays and arrays o

Structures

Reading and Writing Files

Guideline

Some Things to Read

C++ is forwards compatible with C

- If a C++ compiler cannot compile a C program, the C program may be poorly written
- ➤ We will use some basic C++ features in this course, where they make things easier
- One of these ways is file input and output using the iostream standard library.
- Another is the standard string library, but I don't think that Borland C++ 3.1 supports it.

C++ and iostream library An overview of the jost ream library

All Operators and their Precedence

Arrays and Loops

Engineers Nick Urbanik

C Programming for

An overview of the iostream library



Overview of iostream library

To use iostream library in programs, we must include the header file like this:

#include <iostream>

▶ If you are using a very old C++ compiler, put this instead:

#include <iostream.h>

► The library defines three standard stream objects:

std::cin pronounced see-in

an istream class object representing standard input

std::cout pronounced see-out

an ostream class object representing standard output

std::cerr pronounced see-err

an ostream class object representing standard error C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

verview

C++ and iostrea

An overview of the iostream library

Continuing our tour

ata Types

Statements

unctions

ays

nters

Arrays and Pointers

ays and arrays

Structures

Reading and Writing Files

auidelines

Input and Output operators

- Output is done using the left shift operator <</p>
- Input is done using the right shift operator >>>

```
#include < iostream>
int main( void )
       char name[ 1000 ];
       std::cout << "what is your name? ";</pre>
       std::cin >> name;
       if ( name[ 0 ] == '\0' )
               std::cerr ≪ "error: name is empty!\n";
       else
               std::cout ≪ "hello, " ≪ name ≪ "!\n":
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea

An overview of the iostream library

Continuing our tour

ata Types

Statements

uncuons

inters

rave and Poin

Multidimensional

Structures

Structures

Reading and Writing Files

auidelines

How do I remember?

- A way of remembering which operator is which:
- each operator points in the direction the data moves, e.g.,

>> x

puts data into x, while

 $\ll x$

gets data out from x

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream

An overview of the iostream library

Continuing our tour

ata Types

....

rays

ointers

....

rrays and Pointers

iltidimensional rays and arrays o inters

Structures

Reading and Writing Files

Guideline

Why is this better than *printf()* and *scanf()*?

- std::cin, std::cout and std::cerr "know" the type of data they are working with, so
- there is no need for you to remember which format string is needed for which data type;
- "it all just works"

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream

An overview of the iostream library

Continuing our tou

Data Types

_ .

statements

unctions

rays

ointers

.... and Daint

ultidimensional

tructures

Reading and Writing Files

Guideline

Continuing our tour

hello.cpp

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers

Nick Urbanik

hello.cpp

First example

▶ Program hello.cpp:

```
// A first program in C++
#include <iostream>
int main( void )
{
    std::cout << "Hello World\n";
}</pre>
```

▶ Output for Program hello.cpp:

Hello World

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

Continuing our tour

hello.cpp

Basic Synta

Input characters without skipping whitespace

put and from standard ut

eading into strings

.oops Reading till the er

nfinite Loop while reading

ata Types

Statements

Functions

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Good programming practice

- ► Indentation can improves clarity and readability. It can be enhanced by placing braces or blank lines.
- Use variable and function names that explain themselves
- Functions should be shorter than one A4 page and should be simple to understand
- Every long or complicated function should be preceded by a comment describing the purpose of the function.
- Aim: make the program as easy for a human to understand as possible
- Saves money: less time to change/update program = less money.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

library

Continuing our tour

hello.cpp

Basic Syntax

Input characters with skipping whitespace

O of other data to standar utput and from standard

eading into strings

eading till the end of fi

ata Types

Expressions

Statements

Statements

Arrave

Allays

Arrays and Pointers

Multidimensional

tructuroe

Who am I? Who are You Overview

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library
An overview of the iostream

Continuing our tour

Basic Syntax

Input characters without skipping

I/O of other data to standard output

and from standard inportant Reading into strings

Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with ostream

Named Constants

Everesions

Arithmetic Expressions

Precedence

All Operators and their Precedence

Relational Expressions

Logical Expressions

Assignment Expressions
Increment, Decrement Operator

Comma Operator
Arithmetic if Expressions

Bitwise Operators

ements

ple Statements

Scope Scope

while Statemen

Noid Confusing == with =

Jsing a constant or single varia

as a test condition

while and the null statement

for Statement Comparing while and for

if and switch Statements

Functions

Defining Functions
Calling Functions

Calling Functions
Using return Value from Funct

Arrays

Defining Arrays

Arrays and Loops

String

ointers ____

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays o

pointers

Memory Allocation

Multidimensional Arrays

Command Line Argum

Structures

Passing Structures to Function typedef

Reading and Writing Files

fstream: file input and outp Error Handling

Character I/O

getline()

Guidelines

Style Guidelines Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

no am 17 who are ou?

Overview

C++ and iostream

Continuing our tour hello.cpp

Basic Syntax

Input characters without skipping whitespace I/O of other data to standard output and from standard input

eading into stri oops

Reading till the end of file Infinite Loop while readir

ta Types

Statements

Functions

Pointers

Arrays and r omicis

Arrays and arrays pointers

Format of *main()* function without parameters

```
C Programming for
Engineers
```

Nick Urbanik

```
Who am I? Who are You?
```

Overview

C++ and iostream library

Continuing our tour

Basic Syntax

Input characters without skipping whitespace

output and from standa input

eading into strings

.oops

eading till the end of file inite Loop while reading

ata Types

xpressions

Statements

_ .

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

structures

Variable declarations

- Variables are declared and defined with a data type and a name.
- ▶ The name is also called an identifier.
- ► First character of a variable must be letter or underscore (_). Special characters (e.g., \$ and #) are illegal.
- ► Some people (including me!) recommend using lowercase letters and underscores only.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

ello.cpp

Basic Syntax

Input characters without skipping whitespace

I/O of other data to standard output and from standard

iput leading into strings

Loops

Reading till the end of

finite Loop while reading

Data Types

-xpressions

Statements

Otatomonto

rrave

Allays

Arrays and Pointers

Multidimensional

pointers

Continuing our tour

Input characters without skipping whitespace

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers

Nick Urbanik

Input characters without skipping whitespace

Input characters without skipping whitespace

- We can input and output characters one at a time:
- Program one-char-io.cpp:

```
#include <iostream>
int main( void )
{
    char letter;
    std::cin >> letter;
    std::cout << letter;
}</pre>
```

- Note that by default, std::cin >> letter; will skip over whitespace, such as spaces, tabs and newlines
- You can use the iostream member function std::cin.get() to input characters one at a time, including whitespace:

```
#include <iostream>
int main( void )
{
    char letter;
    std::cin.get( letter );
    std::cout << letter;
}</pre>
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrealibrary

ello.cpp asic Syntax

Input characters without skipping whitespace

out eading into strings

Reading till the end of Infinite Loop while read

Data Types

expressions

Statomonte

Functions

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays or pointers

Data Types

Three of the most commonly used data types:

```
int integer
double floating point number
char character
```

▶ Program vars.cpp:

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overviev

C++ and iostream library

iontinuing our to hello.cpp

Basic Syntax

Input characters without skipping whitespace

I/O of other data to standard output and from standard

eading into strings

Loops

Reading till the end of

ata Types

Statements

Arrays

Arrays and Pointers

Multidimensional Arrays and arrays of

The std::endl manipulator

- ► The iostream libraries support *manipulators*
- A manipulator changes the state of a stream
- ► The std::endl manipulator:
 - prints a newline '\n', and
 - sends any remaining characters stored ready for output, to the output
 - We say that this "flushes the buffer"

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overviev

C++ and iostream library

nello.cpp

Basic Syntax

Input characters without skipping whitespace

output and from standar

Loops

Loops

Reading till the end of file infinite Loop while reading

ıta Types

LXPI COSIONO

Statements

Arrave

Delater

Arrays and Pointers

Multidimensional Arrays and arrays of

Continuing our tour

I/O of other data to standard output and from standard input

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers

Nick Urbanik

I/O of other data to standard output and from standard input

Input and output with iostream

- ► Simple input and output operations using iostream library objects std::cout and std::cin
- ▶ Program celcius.cpp:

Output of program celcius.cpp:

```
212 deg F \Rightarrow 100 deg C
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream library

Continuing our tour hello.cop

ello.cpp asic Syntax

skipping whitespace
I/O of other data to standard

output and from standard input

eading into strings

Reading till the end of file Infinite Loop while reading

Data Types

_..p.

Statements

Eupetione

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays o pointers

Using std::cin for input

- std::cin object performs the opposite operation.
- reads text from standard input and assigns to variables
- automatically converts the input text data to appropriate types
- ▶ Program cin-cout-2.cpp:

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

O++ and iostream ibrary

ontinuing our tour

ello.cpp asic Syntax

Input characters withou skipping whitespace

I/O of other data to standard output and from standard input

eading into strings

Reading till the end of file

ata Types

Expression

Statements

Functions

Arrays

Deinter

Arrays and Pointers

Multidimensional Arrays and arrays of

Continuing our tour

Reading into strings

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers

Nick Urbanik

Reading into strings

Strings — array of characters

- ► A *string* is an array of characters, i.e., text.
- ► The length of the string can be defined with a number enclosed in brackets. e.g., array of 10 characters with name *Iname*:

```
char Iname[ 10 ];
▶ Program cin-cout-3.cpp:
  #include <iostream>
  int main( void )
     int num:
     char Iname[ 1000 ];
     std::cout ≪ "Enter class number: ":
     std::cin >>> num;
      std::cout ≪ "Enter last name: ";
     std::cin >> Iname:
      std::cout ≪ "class number is " ≪ num
               \ll ", lastname is " \ll Iname
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream ibrary

hello.cpp

Basic Syntax

skipping whitespace

O of other data to standard output and from standard nput

Reading into strings

Reading till the end of file

Data Types

Expression

Statements

.

riidyo

Arrays and Pointers

Multidimensional Arrays and arrays o

Who am I? Who are You Overview

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library
An overview of the iostream

Continuing our tour

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard output

and from standard input

Loops

Reading till the end of file

Data Types

Characters

ostream
Floating Point Types

Named Constants

Expressions

Arithmetic Expressions
Arithmetic Operators and

All Operators and their Precedence

Relational Expression

Logical Expressions

Increment, Decrement Operato

Comma Operator
Arithmetic if Expressions

Bitwise Operators

tatements

ple Statements

Scope

while Statement

do statement

void Confusing == with = Ising a constant or single variab s a test condition

while and the null statement

Comparing while and for

if and switch Statements

Function

Defining Functions
Calling Functions

Using return Value from Fund

Arrays

Defining Arrays

Arrays and Loops

Strings

ointers

Pointers as Function Paramet

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Arrays of pointers

Memory Allocation

Multidimensional Arrays

Command Line Argumen

Structures

Passing Structures to Function

eading and Writing Files

Error Handling

Character I/O
Reading a Line at a time:

I/O of other data to/from Text File

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers Nick Urbanik

ho am I2 Who ar

Overview

C++ and iostream

ontinuing our tour ello.cpp

asic Syntax

skipping whitespace I/O of other data to standard output and from standard

eading into strings

Loops

Reading till the end of file Infinite Loop while reading

ta Types

. .

Statements

Functions

Pointors

FUILLEIS

Arrays and Pointers

Multidimensional Arrays and arrays of

Loops

```
A loop will cause statements to be executed repeatedly
  until a test condition proves false
```

```
▶ Program while-1.cpp:
```

```
#include <iostream>
int main( void )
       int i = 0;
       while ( j < 5 ) \{
               std::cout \ll "j has the value " \ll j \ll '\n';
               i = i + 1;
```

slide 36/256

Continuing our tour

Reading till the end of file

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers Nick Urbanik

Reading till the end of file

End of file

- When there is no more input, a program has reached the end of file
- When reading standard input:
 - that is redirected from a file (see section 11 on page 19), the program has reached end of file when, well, the last line is read.
 - ► from the keyboard, you can type Control-Z on Windows, or Control-d on Linux to tell your program that it has reached the end of file

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream ibrary

ontinuing our tour ello.cpp

nput characters without kipping whitespace O of other data to stan

input
Reading into strings

oops

Reading till the end of file Infinite Loop while reading

lata Timas

Expressions

Statements

Functions

Arrays

Deinters

Arrays and Pointers

Multidimensional Arrays and arrays of

- ▶ If you execute the code *std*::*cin* ≫ *name*; after reaching end of file, the result is *false*.
- ► That means you can write code like this:

```
if ( std::cin >> name ) {
      // do something with name
}
```

➤ You can also write a loop that automatically terminates when there is no more to read:

► After this loop is finished, either we have reached end of file, or the next input is not a valid **float**

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream
ibrary

Continuing our to hello.cpp

Basic Syntax

skipping whitespace
/O of other data to standard

out eading into strings

oops

Reading till the end of file

minto Loop winto rode

Data Types

Expressions

Statements

T dilottoris

Allays

Assessed Delinter

Multidimensional Arrays and arrays of

Who am I? Who are You

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library
An overview of the iostream

Continuing our tour

hello.cpp

Input characters without skipping

I/O of other data to standard output

and from standard input

Reading into strings

Reading till the end of file

Infinite Loop while reading

Data Types

Characters
Octal, Hexadecimal Output with

ostream Floating Poir

Named Constants

Expressions

Arithmetic Expressions
Arithmetic Operators and

Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions

Assignment Expressions
Increment, Decrement Operator

Comma Operator

Arithmetic if Expressions

Bitwise Operators

tements

ple Statements

Scope Looping Statements

while Statemen

do statement Avoid Confusing =

Jsing a constant or single variates a test condition

while and the null statement for Statement

Comparing while and for if and switch Statements

break, continue, goto

Function

Defining Functions
Calling Functions

Using return Value from Fur

Arrays

Defining Arrays

Arrays and Loops

String:

Pointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Argument

Structures

Passing Structures to Function

ading and Writing Files

fstream: file input and outp

Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text File

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

ho am I? Who are

Overview

C++ and iostream library

ontinuing our tour ello.cpp

ello.cpp lasic Syntax

skipping whitespace I/O of other data to standard output and from standard

put eading into strings

Reading till the end of file Infinite Loop while reading

a Types

_

Functions

Pointers

Arroya and Bai

Arrays and Pointers

Arrays and array pointers

Infinite Loop while reading

If you do something like this:

```
float num = 1.0; std::cout \ll "enter positive floats: "; while ( num > 0 ) { std::cin \gg num; std::cout \ll "You gave me " \ll num \ll ' \n'; }
```

if you enter something that is not a valid **float**, you get an infinite loop.

- Use the method I described in the last slide.
- ► See http://www.parashift.com/c++-faq-lite/input-output.html#faq-15.2 for more about this.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

ontinuing our tour

sic Syntax out characters witho

I/O of other data to standard output and from standard

Reading into strings Loops

Infinite Loop while reading

ata Types

xpressions

Statements

Statements

Arravs

Allays

Arrays and Pointers

Multidimensional

Data Types

- Data is represented in memory by a sequence of bits, arranged into bytes, eight bits to a byte
- ▶ These bits could represent
 - strings or characters
 - integers
 - floating point values
 - memory addresses
 - binary values representing music or video
- Each item of data has a data type
- The data type determines how the program will interpret the data

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

orary

Continuing our tour

Data Types

Characters

Characters

vith ostream

oating Point Types amed Constants

Enumerated Types: enum

expression

Statements

unctions

rrays

Aurent Deleten

Arrays and Pointers

Arrays and arrays o pointers

Structures

Reading and Writing

Declaration & data types

A variable declaration consists of a data type and an identifier, followed by a semicolon: ⟨Data Type⟩ ⟨Variable Name⟩ ⟨semicolon⟩

Example:

int count;

		Data Type
		Integer Type
Basic data types	Keywords	Characters
basic data types	Reywords	Octal, Hexa with ostrear
	•	Floating Pol
character	char	Named Cor
signed character	signed char	Enumerated
•	•	
unsigned character	unsigned char	
integer	int	
•	int	
unsigned integer	IIIL	Arrays
long integer	long	Pointers
unsigned long integer	unsigned long	Arrays and
long long integer	long long	Multidimer
	•	Arrays and
unsigned long long integer	unsigned long long	pointers
short integer	short	
•		Reading a
floating point	float or double or long doubl	e Files
		Guidelines

C Programming for Engineers

Nick Urbanik

Data Types

Integer and floating point data types

Types of integer data (each has an unsigned counterpart):

- - character
 - integer
 - short integer
 - long integer
 - long long integer
- Types of floating point values:
 - single precision (float)
 - double precision (double)
 - long double precision (long double)
- The range of values that can be held are machine dependent.
- ▶ The ranges of integer types are in limits.h
- ► The ranges of floating types are in float.h

C Programming for Engineers

Nick Urbanik

Data Types

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

tinuing our to

hello.cpp

Input characters without skipping

I/O of other data to standard output

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types

Characters Octal, Hexadecimal Output with

Floating Point Types Named Constants

Enumerated Types: enur

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions

Assignment Expressions
Increment, Decrement Operate

Comma Operator

Arithmetic if Expressions Bitwise Operators

Casts

Statements

nple Statements

Scope Looping Statements

while Statemen

Avoid Confusing == with Using a constant or single

as a test condition

while and the null statement for Statement

Comparing while and for fand switch Statements

Function

Defining Functions
Calling Functions

Using return Value from I

Arrays

Defining Arrays

Arrays and Loops

String:

Pointers

Pointers as Function Parameter

ays and Pointer

and pointers

Multidimensional Arrays and arrays of

pointers

Memory Allocation

Multidimensional Arrays

Command Line Argumen

Structures

Passing Structures to Function

Reading and Writing Files

fstream: file input and out Error Handling

Character I/O
Reading a Line at a time:

I/O of other data to/from Text File

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers Nick Urbanik

NICK Urbanik

no am 17 who are ou?

Jverview

C++ and iostream library

Continuing or

ta Types

Integer Types

Octal, Hexadecimal Output with ostream

Floating Point Types Named Constants

Enumerated Types: enum

tatomonte

rrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing

auidelines

Signed integers — 1

- An unsigned integer has only positive values while a signed integer can have a positive or negative value.
- Signed Integers:

int	use: standard integer
size:	system dependent, usually size of a word.
range:	INT_MIN to INT_MAX, defined in limits.h. For 4 byte word, $-2^{31} - 1$ to 2^{31} , i.e., -2147483648 to 2147483647
example declaration:	int num;
example constant:	1000

C Programming for Engineers

Nick Urbanik

Who am I? Who are

verview

++ and ios rary

to Tunco

Integer Types

Octal, Hexadecimal Output with ostream

Floating Point Types Named Constants Enumerated Types: **enum**

prossions

latomonto

Arrays

Pointers

Arrays and Pointe

rrays and an ointers

Structures

eading and Writing

iidelines

Signed integers — 2

example declaration:

long use: large numbers

size: usually 4 bytes.

LONG MIN to LONG MAX. For 4 range:

bytes. -2^{31} to $2^{31} - 1$, i.e., -2147483648 to 2147483647

long Inum;

example constant: 5212000L

short use: smaller numbers

2 bytes or same size as integer size:

SHRT_MIN to SHRT_MAX. For 2 bytes, -2¹⁵ range:

to $2^{15} - 1$, i.e., -32768 to 32767

declaration: short snum:

120 constants:

C Programming for Engineers

Nick Urbanik

Integer Types

slide 47/256

long long

long long use: very large numbers size: usually 8 bytes. for 8 bytes. -2^{63} to $2^{63} - 1$, i.e., range: -9.223.372.036.854.775.808 to 9.223.372.036.854.775.807 example declaration: long long *lnum*; example constant: 5212000LL unsigned long long use: very large positive numbers usually 8 bytes. size: for 8 bytes, 0 to $2^{64} - 1$, i.e., 0 to range:

5212000LL

18.446.744.073.709.551.615

unsigned long long lnum;

C Programming for Engineers Nick Urbanik

Integer Types

slide 48/256

example declaration:

example constant:

Unsigned integers — 1

unsigned int

size: system dependent; always same size as int

range: 0 to *UINT_MAX*. For 4 byte word, 0 to

 $2^{32} - 1$, i.e., 4294967295

declaration: **unsigned int** *unum*; or **unsigned** *unum*;

constants: 5530u

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

O<mark>++ and</mark> iostream ibrary

- -

Integer Types

Observations

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants

Enumerated Types: e

tatements

trravs

rrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing Files

auidelines

Unsigned integers — 2

unsigned long

size: usually 4 bytes; always same size as **long**

range: 0 to UINT_MAX. For 4 byte, 0 to

 $2^{32} - 1 = 4294967295$

example declaration: **unsigned long** *ulnum*;

example constant: 76212000uL

unsigned short

size: usually 2 bytes; always same size as **short**

int

range: 0 to USHRT_MAX. For 2 bytes, 0 to 65536

example declaration: **unsigned short** *usnum*;

example constant: 34000u

C Programming for Engineers

Nick Urbanik

Who am I? Who are

verview

++ and iostrea rary

. =

Integer Types

Characters

with ostream
Floating Point Types

Floating Point Types
Named Constants
Enumerated Types: enum

omonte

llements

unctions

ırrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays pointers

Structures

Reading and Writing Files

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

tinuing our to

hello.cpp

Input characters without skipping

Whitespace
I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Typ

Characters
Octal, Hexadecimal Output with

ostream
Floating Point Types
Named Constants

Named Constants Enumerated Types: enum

Expressions

Arithmetic Expressions Arithmetic Operators and All Operators and their Precedence

Relational Expression onical Expressions

Assignment Expressions

Increment, Decrement Operator

Arithmetic if Expressions Bitwise Operators

Casts

tatements

ple Statements

Scope Looping Statements

while Statemer

Avoid Confusing == with =
Jsing a constant or single varia

is a test condition

while and the null statement

or Statement Comparing while and for

f and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers of Eurotian Paramete

rays and Pointers

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Arguments:

Structures

Passing Structures to Function

leading and Writing Files

Error Handling Binary files

Reading a Line at a time:

I/O of other data to/from Tex

Style Guide

Program Do Modules

Some Things to Rea

C Programming for Engineers

Nick Urbanik

/ho am I? Who are

Overview

C++ and iostream library

ontinuing our tour

a Types

Characters

with ostream
Floating Point Types

Named Constants
Enumerated Types: enum

Expressions

Statements

...

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing

Characters

- ▶ When working with characters, use the type **char**.
- Note that the type char can be signed or unsigned, depending on the compiler.

char

size: usually 8 bits; CHAR_BIT in

limits.h

range: CHAR_MIN to CHAR_MAX. For 1

byte, could be -2^7 to $2^7 - 1$, i.e.,

−127 to 128 *or* 0 to 255

example declaration: char ch;

example constant: 'a'

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostre brary

ntinuing our toui

ata Types nteger Types

Characters

with ostream
Floating Point Types
Named Constants
Fnumerated Types: enum

жргооологго

ilements

Arrays

Tirays

Pointers

Arrays and Pointers

Arrays and arrays pointers

Structures

Reading and Writing

Characters: signed, unsigned

specify the type as signed or unsigned only if you care.

signed char

size: usually 8 bits; CHAR_BIT in limits.h

range: SCHAR_MIN to SCHAR_MAX. For 1 byte,

 -2^7 to $2^7 - 1$, i.e., -127 to 128

example declaration: signed char ch;

example constant: 'a'

unsigned char

size: usually 8 bits; CHAR_BIT in limits.h

range: 0 to UCHAR_MAX. For 1 byte, 0 to

 $2^8 - 1 = 255$

example declaration: **unsigned char** *ch*;

C Programming for Engineers

Nick Urbanik

Vho am I? Who are ou?

verview

++ and iostream orary

ntinuing our tour

Integer Types
Characters

Octal, Hexadecimal Output

Floating Point Types
Named Constants

Apr 03310113

otatements

rravs

..ayo

Arrays and Pointers

Multidimensional Arrays and arrays opointers

Structures

eading and Writing les

Character types

- Characters are represented in C with integer values.
- ► The correspondence between a given character and an integer value is determined by an agreed-upon character set, such as the ASCII character set.
- Examples of declarations:

```
char letter;
signed char sletter;
unsigned char uletter;
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overviev

++ and iostreaderary

ontinuing our tour

Integer Types

Characters
Octal, Hexadecimal Output

with ostream
Floating Point Types
Named Constants

Enumerated Types: er

.

tatements

rrave

lays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays o pointers

Structures

Reading and Writing

Program princhar.cpp

```
#include < iostream>
int main( void )
   char letter = 'A'; // 'A' is character constant
   int num = letter;
   char ch1 = 'b'; // ASCII code = 98
   char ch2 = 'B'; // ASCII code = 66
    std::cout << "letter = " << letter
        \ll ", num = " \ll num \ll '\n':
    std::cout \ll "letter + 1 = " \ll letter + 1
        \ll ", num = " \ll num \ll '\n':
    std::cout \ll ch1 \ll " - " \ll ch2
        \ll " = " \ll ch1 - ch2 \ll std::endl;
Output for Program princhar.cpp:
letter = A, num = 65
letter + 1 = 66, num = 65
b - B = 32
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

)verview

++ and iostream

ontinuing our tour

Data Types Integer Types

Characters
Octal, Hexadecimal Ou

Floating Point Types
Named Constants

-xpi e5510115

olalements

rravs

..., 5

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing Files

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

itinuing our to

hello.cpp Rasic Synt

Input characters without skipping

I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators and All Operators and their Precedence

Relational Expressions

Assignment Expressions

Increment, Decrement Operato

Arithmetic if Expressions

Bitwise Operator:

statements

ments

ole Statements

Looping Statement

while Statement

void Confusing == with =

Jsing a constant or single variable

a a test condition

while and the null statement

Comparing while and for

if and switch Statements

Function

Defining Functions
Calling Functions

Calling Functions
Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers as Eupation Paramete

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Arrays of pointers

Memory Allocation

Multidimonsional Arrays

Command Line Argument

Structures

Passing Structures to Function

Reading and Writing Files

Error Handling
Binary files
Character I/O

Reading a Line at a time:

I/O of other data to/from Text Fi

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

no am I? Who are

Dverview

C++ and iostream library

Continuing our tour

ta Types

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants

xpressions

Statements

...

Pointore

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing

Integer constants: octal and hexadecimal

An integer can be represented in an octal or a hexadecimal form. Octal integer constants are represented with a leading zero. Hexadecimal integer constant is represented with the leading characters 0x, or 0X.

Integer	Octal	Hexadecimal
4	04	0x4
12	014	0xc
123	0173	0x7b

- The long integer qualifier L can also used with the octal and hexadecimal
- Example of octal constant: 0553000L
- Example of hexadecimal constant: 0x2f6c7a333L

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

++ and iostream brary

Continuing our to

Integer Types

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants
Enumerated Types: enum

Statements

rravs

iiays

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing Files

Program printvar.cpp

```
#include < iostream>
#include < iomanip>
int main( void )
   int num = 77;
   short int small = 0173:
   short little = 0x7b:
   long int big = 88000;
   long large = -43000L:
   unsigned int unum = 45000;
   unsigned long ubig = 33000000UL;
    std::cout << std::showbase:
    std::cout << "num (dec) = " << num</pre>
        << ", (oct) = " << std::oct << num</pre>
        \ll ", (hex) = " \ll std::hex \ll num \ll '\n';
    std::cout << "small (oct) = " << std::oct << small
        \ll ", little (hex) = " \ll std::hex \ll little \ll '\n';
    std::cout << "big (dec) = " << std::dec << big
        << ", large (dec) = " << large << '\n':</pre>
    std::cout << "unum = " << unum
        \ll ", ubig = " \ll ubig \ll '\n';
    std::cout << "small (dec) = " << small
        << ", little (dec) = " << little << std::endl;</pre>
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

ontinuing our tour

nteger Types

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants
Enumerated Types: enum

tatomonts

.....

iiays

ointers

Arrays and Pointer

Multidimensional Arrays and arrays o pointers

Structures

Reading and Writing Files

Output for Program printvar.cpp

```
num (dec) = 77, (oct) = 0115, (hex) = 0x4d small (oct) = 0173, little (hex) = 0x7b big (dec) = 88000, large (dec) = -43000 unum = 45000, ubig = 330000000 small (dec) = 123, little (dec) = 123
```

▶ We need to include iomanip here for the *manipulators*:

```
std::oct Changes the state of the ostream to displaying all integer type numbers in octal.

All numbers printed while in this state are in octal.
```

std::hex Changes the state of the ostream to displaying all integer type numbers in hexadecimal

std::dec Changes the state of the ostream to displaying all integer type numbers in decimal

std::showbase a state that displays octal with leading "0", hexadecimal with leading "0x"

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ontinuing our to

nteger Types haracters

Octal, Hexadecimal Output with ostream

Named Constants
Enumerated Types: enum

Statements

rravs

Pointers

Arrays and Pointer

lultidimensional rrays and arrays ointers

Structures

eading and Writing les

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

tinuing our to

hello.cpp Basic Synt:

Input characters without skipping

whitespace
I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types

ctal, Hexadecimal Output with

Floating Point Types

Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators and All Operators and their Precedence

Relational Expressions

Assignment Expression

Increment, Decrement Operato

Comma Operator

Bitwise Operators

totomonto

tatements

ple Statements

Scope Looning Statements

while Statemer

do statement Avoid Confusing ==

Using a constant or single varia as a test condition

while and the null statement

Comparing while and for

f and switch Statements

Function

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

rays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

ointers

Memory Allocation
Multidimensional Arrays

Command Line Argumen

Structures

Passing Structures to Function

Reading and Writing Files

fstream: file input and out Error Handling

Binary files Character I/O

Reading a Line at a time: getline()

I/O of other data to/from Text FI

Guidelines

Program Design

Some Things to Re

C Programming for Engineers

Nick Urbanik

io am I? Who are

Overview

C++ and iostream library

Continuing our tour

ta Types

Characters
Octal, Hexadecimal Output

with ostream
Floating Point Types

Named Constants

Expressions

Statements

rrave

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing

auidelines

Floating point data types — 1

The floating point data type is used to represent real numbers. Real numbers include the fractional number between integers. two components: an exponent and a fraction

float	t
-------	---

size: system dependent, usually four

bytes.

range: FLT_MIN to FLT_MAX defined

in float.h

example declaration: **float** *fnum*;

example constant: 3.456f

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostrea ibrary

ontinuing our tour

nteger Types Characters Octal, Hexadecimal Output

Floating Point Types

Named Constants

Fnumerated Types: enum

_

Otatomonts

rrays

. . .

Arrays and Pointers

Multidimensional Arrays and arrays pointers

Structures

Reading and Writing

Floating point data types — float

double

size: twice the size of float, usually eight

bytes

range: DBL_MIN to DBL_MAX defined in

float.h

example declaration: double dnum;

example constants: floating point notation: 3.4567

exponential notation: 4.788e+5, 3e1

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream brary

ontinuing our tour

Integer Types Characters

Octal, Hexadecimal Out with ostream

Floating Point Types
Named Constants

Enumerated Types: enum

Ctatamanta

Statements

unctions

Arrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing Files

Floating point data types — long double

long double

size: bigger than the size of double.

range: LDBL_MIN to LDBL_MAX defined in

float.h

example declaration: long double ldnum;

example constants: 3.4567L exponential notation:

4.788e+5L

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

Continuing our tour

Integer Types Characters

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants
Frumerated Types: enum

Enumerated Types: enum

Statomonte

Statements

Δrrave

irays

Name of Deliates

Multidimensional

0.....

Reading and Writing

auidelines

Program float-io.cpp

```
#include < iostream>
int main( void )
       float cost = 15.92;
       float total = 3.6e5:
       float value = 357e-1: // value is 35.7
       double debt = 1.2e15;
       long double decrease = 5e-6;
       std::cout << "cost = " << cost
                \ll ", total = " \ll total \ll '\n':
       std::cout << "value = " << value << '\n';</pre>
       std::cout << "debt = " << debt << '\n'
                 \ll "decrease = " \ll decrease \ll ' \n':
Output for Program float-io.cpp
cost = 15.92, total = 360000
value = 35.7
debt = 1.2e + 1.5
decrease = 5e-06
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overviev

C++ and iostream

ontinuing our tour

nteger Types Characters

Octal, Hexadecimal Output with ostream

Floating Point Types
Named Constants

Expressions

Statements

Otatorionio

..., -

Arrays and Pointers

Multidimensional

pointers

Structure

Reading and Writing Files

auidelines

Program float.cpp showing limits of floating values

```
#include <float.h>
#include < iostream>
int main()
        std::cout << "float min: " << FLT_MIN << '\n':
        std::cout << "float max: " << FLT_MAX << '\n';</pre>
        std::cout ≪ "double min: " ≪ DBL_MIN ≪ '\n';
        std::cout << "double max: " << DBL_MAX << '\n';</pre>
        std::cout \ll "long double min: " \ll LDBL_MIN \ll ' \n';
        std::cout ≪ "long double max: " ≪ LDBL_MAX ≪ '\n';
        std::cout ≪ "float epsilon: " ≪ FLT_EPSILON ≪ '\n';
        std::cout \ll "double epsilon: " \ll DBL\_EPSILON \ll ' \n';
        std::cout \ll "long double epsilon: " \ll LDBL\_EPSILON \ll '\n'; std::cout \ll "A long double constant: " \ll 4.788e+5L \ll '\n'; Ardys and Pointers
        return 0:
```

C Programming for Engineers

Nick Urbanik

Floating Point Types

Output for Program float.cpp

```
float min: 1.17549e-38
float max: 3.40282e+38
double min: 2.22507e-308
double max: 1.79769e+308
long double min: 3.3621e-4932
long double max: 1.18973e+4932
float epsilon: 1.19209e-07
double epsilon: 2.22045e-16
long double epsilon: 1.0842e-19
A long double constant: 478800
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream library

Continuing our tour

Integer Types

Octal, Hexadecimal Output

Floating Point Types
Named Constants

Named Constants Enumerated Types: enum

Lxpressions

Statements

Functions

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing

auidelines

Data Types

Named Constants

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers

Nick Urbanik

Named Constants

Symbolic constants: #define, const, and enum

- #define can be thought of as a global substitution command.
- The #define directive consists of the #define keyword, a define symbol, and a replacement text.
- ▶ Program define.cpp:

#include < iostream>

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overviev

C++ and iostream brary

Continuing our tour

nteger Types characters

Octal, Hexadecimal Output with ostream

Named Constants

Enumerated Types: enum

Litariorated Types. enam

Statements

Tunationa

rays

nintare

Arrays and Point

Multidimensional Arrays and arrays opointers

Structures

Reading and Writing

Guidelines

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ntinuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard output

and from standard inpu

Reading into strings Loops

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output with

Floating Point

Named Constants

Enumerated Types: enum

Arithmetic Expressions
Arithmetic Operators and

All Operators and their Precedence

Relational Expressions

Logical Expressions

Assignment Expressions
Increment Decrement Operator

Comma Operator
Arithmetic if Expressions

Bitwise Operator

Statements

ple Statements

Scope Looning Statement

while Statemen

do statement Avoid Confusing =

Using a constant or single varia as a test condition

while and the null statement

Comparing while and for

f and switch Statements

Functions

Defining Functions

Calling Functions
Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

String

Pointers

Pointers as Function Parameter

ays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Arguments

Structures

Passing Structures to Function

Reading and Writing Files

fstream: file input and out Error Handling

Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text F

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

no am I? Who are

Overview

C++ and iostream library

Continuing our tour

ta Types

Characters
Octal, Hexadecimal Output

with ostream
Floating Point Types

Enumerated Types: enum

Expressions

Statements

Arravs

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing

auidelines

Kconst and enum

▶ Using const is better than #define. Program const.cpp:

```
#include < iostream>
int main( void )
   const float rate = 1.5: // with ":"
   float cost = rate * 8:
    std::cout << "cost = " << cost
        << ". rate = " << rate << '\n':</pre>
  Program enum-2.cpp:
#include < iostream>
enum weather { clouds, rain, sunny, storm };
int main( void )
       weather today = sunny;
        std::cout << clouds << ' ' << rain << ' '
            \ll sunny \ll ' ' \ll storm \ll '\n';
        std::cout << "today: " << today << '\n';</pre>
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ontinuing our tour

ata Types
nteger Types
Characters

oth ostream loating Point Types

Enumerated Types: enum

Oldioi i i o i i o

rravs

ointers

rrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing Files

Guidelines

Enumerated types — 2

▶ Program enum-3.cpp:

```
#include < iostream>
enum weather \{ clouds = 30, rain = 5, 
               sunny = 255, storm = 1 };
enum boolean { TRUE = 1, FALSE = 0 };
int main( void )
    weather today = rain;
    std::cout \ll clouds \ll ' ' \ll rain \ll ' '
             \ll sunny \ll ' ' \ll storm \ll '\n';
    std::cout \ll "today: " \ll today \ll ' \n';
```

► Output of program enum-3.cpp:

30 5 255 1

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream library

ontinuing our tour

Integer Types

Octal, Hexadecimal Output with ostream

Named Constants

Enumerated Types: enum

Ctatamanta

Otatomonio

Arravs

..., -

.

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

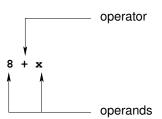
Structures

Reading and Writing Files

Guidelines

Expressions

- Expressions define the tasks to be performed in a program do calculations perform function calls and assignment operations
- Arithmetic expressions perform the standard arithmetic operations.
 - Arithmetic operator is a symbol or a binary operator, i.e. acts upon 1 or more operands or values.
 - Operand may be constant, variable, or function.



C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostrea orary

ontinuing our to

ata Types

Expressions

Arithmetic Expressions
Arithmetic Operators and
Precedence

All Operators and their

Relational Expression

Increment, Decrement Operators Comma Operator

Arithmetic if Express
Bitwise Operators

tatements

Otatomonto

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays o

Expressions

Arithmetic Expressions

All Operators and their Precedence

Arrays and Loops

C Programming for Engineers Nick Urbanik

Arithmetic Expressions

slide 73/256

Arithmetic operators — unary

- Unary arithmetic operators:
 - + plus, or positive
 - negative

```
-6 negative 6

-(-6) positive 6

-(3-7) positive 4
```

change the sign of operand *y*

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream ibrary

Jonanian g our

ata Types

Arithmetic Expressions

rithmetic Operators and

All Operators and the Precedence

Relational Expressions Logical Expressions

Assignment Expression Increment, Decrement

Operators Comma Operator

Arithmetic if Expres Bitwise Operators

.....

Statements

Arrays and Pointers

Multidimensional arrays and arrays of

Arithmetic operators — binary

Additive operators:

```
+ addition
```

subtraction

```
total = 8 + 5 - 2;

sum = num - 3;
```

► Multiplicative operators:

```
* multiplication
```

/ division

% modulo or remainder

```
int total;
total = 8 * 5;
total = 23 / 4; // now total has the value 5
total = 23 % 4; // now total has the value 3
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream ibrary

Continuing our tou

Data Types

Arithmetic Expressions

rithmetic Operators and recedence

All Operators and thei Precedence

Logical Expressions

Assignment Expression Acrement, Decrement Operators

Comma Operator Arithmetic if Expres

Jasts

tatements

Arrays

Officia

Arrays and Pointers

uitidimensional rrays and arrays of

Who am I? Who are You Overview Quick Tour Standard Input, Stan

+ and iostream library
An overview of the iostream

library

hello.cpp Basic Syntax Input characters without skipping

whitespace I/O of other data to standard output

and from standard inp Reading into strings

Reading till the end of file

Data Types

Characters
Octal, Hexadecimal Output with

Floating Point Types

Named Constants
Enumerated Types: enum

Expressions

Arithmetic Expressions

Arithmetic Operators and Precedence

All Operators and their Precedence

Relational Expressions ogical Expressions

Assignment Expressions Increment, Decrement Operator

Comma Operator
Arithmetic if Expressions

Bitwise Operators

tatements

ple Statements

Scope Looping Statements

while Statement do statement

woid Confusing == with = Ising a constant or single variab s a test condition

while and the null statement

Comparing while and for

f and switch Statements

Functions

Calling Functions

Using return Value from

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Parameter

rrays and Pointer

Strong relationship between array and pointers

ultidimensional Arrays and arrays of

Arrays of pointer

Memory Allocation
Multidimensional Arrays

Command Line Argume

Structure

Passing Structures to Functio

eading and Writing Files

fstream: file input and outp

Binary files
Character I/O
Reading a Line at a tim

getline()
I/O of other data to/from Text F

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

/ho am I? Who are ou?

Dverview

C++ and iostream

ontinuing our tour

ata Types

Expressions

Arithmetic Operators and Precedence

All Operators and their

Precedence Relational Expressions

Assignment Expressions Increment, Decrement

Operators
Comma Operator
Arithmetic if Express

Arithmetic if Expres Bitwise Operators

tatomonte

Otatomonto

Arrays

Arrays

Pointers

rrays and Pointers

viulitaimensional Arrays and arrays o

Arithmetic precedence and associativity

- One operator may take precedence over another.
- ▶ When an expression is evaluated, it is broken down into a series of subexpressions, one for each operator.
- ► The order in which these subexpressions are evaluated is determined by either parentheses or precedence.
- Example:

$$(2+4)*(5-3)$$

evaluates to the same as

- If there are no parentheses, precedence determines the order of evaluation
- ▶ All operators are ranked according to their precedence.
- Operators with greater precedence are evaluated first.
- Example:

$$2 + 4 * 5 - 3$$

evaluates to the same as:

$$2 + 20 - 3$$

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostr rary

oto Timos

ta Types

Arithmetic Expressions

Arithmetic Operators and Precedence

All Operators and their Precedence

Assignment Expressions Increment, Decrement Operators

Arithmetic if Expressi Bitwise Operators

tatomonte

Statements

Functions

Arrays

Pointers

Arrays and Pointer

rrays and arrays (

Arithmetic Precedence and Associativity (cont')

- If the operators share the same operand, the priority of one operator over the other will be determined by its associativity. Associativity describes how an operator associates its operands.
- Arithmetic operators associate left to right, whereas assignment operators associate right to left.
- Example:

$$2 + 10 - 5$$

evaluates the same as

$$12 - 5$$

The following table lists the precedence and assciativity of operators. C Programming for Engineers

Nick Urbanik

Who am I? Who are fou?

Overview

++ and iostrea orary

ontinuing our to

ta Types

Arithmetic Expressions
Arithmetic Operators and

Arithmetic Operators and Precedence

Relational Expressions Logical Expressions Assignment Expressions Increment, Decrement Operators Comma Operator

Bitwise Operators Casts

Statements

Functions

Arrays

Pointers

Arrays and Pointe

viuitidimensional Arrays and arrays

Who am I? Who are You Overview

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

library

ontinuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard output

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

ostream Floating Poir

Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators an Precedence

All Operators and their Precedence

Relational Expressions

Logical Expressions Assignment Expression

Increment, Decrement Operator

Arithmetic if Expressions

Bitwise Operators Casts

tatements

ple Statements

Scope Looping Statement

while Statement

Avoid Confusing == with =

Jsing a constant or single varia

as a test condition

while and the null statement

Comparing while and for

f and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Funct

Arrays

Defining Arrays

Arrays and Loops

String:

Pointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

ultidimensional Arrays and arrays of

pointers

Memory Allocation Multidimensional Arrays

Command Line Argume

Structures

Passing Structures to Function typedef

eading and Writing Files

Error Handling Binary files

Reading a Line at a time:

I/O of other data to/from Text F

auldelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

ho am I? Who are u?

Overview

C++ and iostrea

Continuing our tour

ata Types

Expression

Arithmetic Expressions
Arithmetic Operators and

All Operators and their Precedence

Relational Expression:

Assignment Expressions

Comma Operator
Arithmetic if Express

Bitwise Operato

Statements

_ ..

Arrays

7 11 Tay 0

1 Ulliters

Arrays and Pointers

Arrays and arrays of

All Operators and their Precedence I

Level	Operator	Function
15L	->, . []	structure member selectors array index
	()	function call
14R	sizeof	size in bytes
	++,	increment, decrement
	\sim	bitwise NOT
	!	logical NOT
	+, -	unary plus, minus
	*, &	dereference, address-of
	()	type conversion (cast)
13L	*, /, %	multiply, divide, modulus
12L	+, -	arithmetic operators

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and i

ontinuing our to

ta Types

ynressini

rithmetic Expressions rithmetic Operators and

All Operators and their Precedence

Logical Expressions
Assignment Expression
Increment, Decrement
Operators

Somma Operator Arithmetic if Expres Bitwise Operators

latements

.

Arrays

Casts

Arrays and Pointers

iuitidimensional rrays and arrays of

All Operators and their Precedence II

Level	Operator	Function
11L	≪, ≫	bitwise shift
10L	<, <=, >, >=	relational operators
9L	==, !=	equality, inequality
8L	&	bitwise AND
7L	^	bitwise XOR
6L		bitwise OR
5L	&&	logical AND
4L		logical OR
3L	?:	arithmetic if
2R	=	assignment operator
	*=, /=, %=	compound assignment operators
	+=, -=, <<= >>=, &=, =, ^=	
1L	, x=, =, =	comma operator

C Programming for Engineers

Nick Urbanik

/ho am I? Who are

)verview

++ and iostream orary

ila Types

Arithmetic Expressions

Precedence
All Operators and their

All Operators and their Precedence

Logical Expressions
Assignment Expressions
Increment, Decrement
Operators
Comma Operator
Arithmetic if Expressions

asts

atements

Functions

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Who am I? Who are You

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ntinuing our to

hello.cpp

Input characters without skipping

I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

ostream Floating Poin

Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence Relational Expressions

Logical Expressions

Assignment Expressions Increment Decrement Operator

Comma Operator

Bitwise Operators

tatements

ple Statements

Scope

while Statement

No statement Avoid Confusing == with = Using a constant or single varial

as a test condition

while and the null statement

for Statement

Comparing while and for if and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Func

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Parameter

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Memory Allocation

Command Line Arguments: ai

Structures

Passing Structures to Function

eading and Writing Files

Error Handling
Binary files
Character I/O

Reading a Line at a time:

I/O of other data to/from Text F

auldelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

no am i? who are u?

Jverview

C++ and iostream

Continuing our tour

ata Types

pressions

rithmetic Expressions rithmetic Operators and

All Operators and their

Relational Expressions

Logical Expressions
Assignment Expressions
Increment, Decrement

Comma Operator Arithmetic if Express

Casts

Statements

.

Arrays

Pointers

Arrays and Pointers

viulitaimensional Arrays and arrays o

Comparison Expressions

- Relational, equality, and logical expressions compare their operands. the result of the comparison is the integer value of either one or zero.
- If an operation compares successfully, the result of the expression is an integer value of 1. If an operation compares fails, the result of the expression is an integer value of 0.
- A relational operator compares two operands and determines whether one is greater or less than the other.
 - < less than
 - greater than
 - <= less than or equal
 - >= greater than or equal

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

rary

Jonanaing of

ata Types

Expressions
Arithmetic Expressions

All Operators and their Precedence

Relational Expressions

Logical Expressions
Assignment Expressi
Increment Decrement

Operators
Comma Operator
Arithmetic if Expression
Bitwise Operators

atements

Arrays

Pointers

Arrays and Pointe

lultidimensional rrays and arrays (

Relational expressions — 1

- Relational expressions can be combined with other expressions.
- Example:

```
num = 3;

abc = (num < 5);
```

As 3 < 5, the resulting value of *abc* is 1.

► Example:

```
num = 8;

abc = 5 + (num < 5);
```

As 8 > 5, the value of *abc* is 5 + 0 = 5.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream brary

onlinuing our li

ata Types

Expressio

Arithmetic Operators and

Precedence

Relational Expressions

Assignment Expressions
Increment, Decrement
Operators
Comma Operator
Arithmetic if Expressions

Casts

Statements

Λ.....

Arrays

Arrays and Poin

Multidimensional Arrays and arrays o

Relational expressions — 2

- The equality operators test a relationship between two operands
- result is 1 or 0.

```
== equal,
!= not equal
```

- equality operator is a double equal sign ==
- assignment operation is a single equals sign = e.g

```
while (test == 1) // comparison operation for equality
test = 1;
                            assignment operation
```

C Programming for Engineers

Nick Urbanik

Relational Expressions

Expressions

All Operators and their Precedence

Logical Expressions

Arrays and Loops

C Programming for Engineers

Nick Urbanik

Logical Expressions

Logical expressions — 1

- The logical operators compare the truth or false of their operands. determined by whether or not it has a zero value
 - If an expression evaluates to zero, the expression is false
 - If an expression evaluates to a non-zero, it is true.
- ► The operands of a logical operation are often relational expressions.
 - logical AND &&
 - logical OR
 - logical NOT

C Programming for Engineers

Nick Urbanik

Logical Expressions

Logical expressions — 2

► Here is the truth table for the logical AND operation:

expr1		expr2		(expr1) && (expr2)	
Т	non-zero	Т	non-zero	Т	1
T	non-zero	F	0	F	0
F	0	Т	non-zero	F	0
F	0	F	0-zero	F	0

► Truth table for the logical OR operation:

expr1		expr2		(expr1) (expr2)		
Т	non-zero	Т	non-zero	Т	1	
T	non-zero	F	0	Т	1	
F	0	Т	non-zero	Т	1	
F	0	F	0-zero	F	0	

Truth table for the logical NOT operation:

	expr	!(expr)		
Т	non-zero	F	0	
F	0	Т	1	

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

++ and iost orary

ita Types

Expression

Arithmetic Operators and

All Operators and the Precedence

Relational Expressions

Logical Expressions

Increment, Decrement Operators Comma Operator Arithmetic if Expression

Bitwise Operators

tatomonte

Otatomonto

A

Arrays

Pointers

Arrays and Pointers

viulitaimensional Arrays and arrays o

Program countdig.cpp

```
/* Counts only the numeric characters '0' - '9'
   read from standard input */
#include < iostream>
int main( void )
    int n = 0:
    char c:
    while ( std::cin \gg c ) {
        if (c > = '0' \&\& c < = '9')
            n = n + 1:
    std::cout \ll "Count of digits = " \ll n \ll ' \n';
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

Continuing our to

ata Types

Express

Arithmetic Expressions

recedence

Relational Expression

Logical Expressions

Assignment Expression

Operators
Comma Operator
Arithmetic if Expressions

Statamenta

Statements

Arrovo

Arrays

Arrays and Poin

Multidimensional Arrays and arrays of

Who am I? Who are You

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ontinuing our to

hello.cpp

Input characters without skipping

I/O of other data to standard output

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

Floating Point Types Named Constants

Named Constants
Enumerated Types: enum

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

Relational Expressions

gical Expressions

Assignment Expressions

ncrement, Decrement Operator

Arithmetic if Expressions Bitwise Operators

Bitwise Operators Casts

Statements

ple Statements

Scope Looping Statements

while Statement

o statement woid Confusing == with = Using a constant or single variab

as a test condition

while and the null statement

for Statement Comparing while and for

if and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Functi

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Parameter

rays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Argumer arav

Structures

Passing Structures to Function typedef

ading and Writing Files

Error Handling Binary files

Reading a Line at a time:

I/O of other data to/from Text F

Guideline

Program Design

Some Things to Read

C Programming for Engineers

Nick Urbanik

no am 17 wno are ou?

Jverview

C++ and iostrea library

Continuing our tour

ata Types

kpressions

ithmetic Expressions ithmetic Operators and

All Operators and their Precedence

Relational Expressions Logical Expressions Assignment Expressions

Increment, Decrement Operators Comma Operator Arithmetic if Expressions Bitwise Operators

Statements

.

Arrays

Pointers

Arrays and Pointers

rrays and arrays of

Assignment expressions

- assignment operation is an expression.
- resulting value of the assignment expression is the value assigned to the variable in the assignment operation.
- assignment operation can be combined with other operators to form a complex expression:

```
total = (num = (4 + 2));
```

Parentheses can be left out since assignment evaluate from right to left.

```
total = num = 4 + 2;
```

which is identical to

```
total = 4 + 2;

num = 4 + 2;
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

++ and iostrea orary

ontinuing our t

ata Types

expressions

rithmetic Operators and

All Operators and the Precedence

Relational Expression Logical Expressions

Logical Expressions
Assignment Expressions

Increment, Decrement Operators Comma Operator Arithmetic if Expressions Bitwise Operators

atements

Functions

Arrays

Pointers

Arrays and Pointers

Multidimensional arrays and arrays of

Arithmetic assignment operators

 provide a shorthand applying an arithmetic operation to a variable

```
    j += 3; is equivalent to j = j + 3;
    j *= 3; is equivalent to j = j * 3;
```

- List of arithmetic assignment operators:
 - += add and then assign
 - -= subtract and then assign
 - *= multiply and then assign
 - /= divide and then assign
 - %= modulo; assign remainder

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream orary

onlinuing our to

ata Types

Expression

Arithmetic Expressions

Il Operators and their

Relational Expressions

ogical Expressions

Assignment Expressions Increment, Decrement Operators

Comma Operator
Arithmetic if Expressio

Casts

tatements

_ ..

Arrays

. . .

Arrays and Pointers

lultidimensional rrays and arrays of

Who am I? Who are You

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

Continuing our to

hello.cpp

Input characters without skipping

I/O of other data to standard output

and from standard inp

Reading into strings Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

Floating Point Types Named Constants

Named Constants
Enumerated Types: enur

Expressions

Arithmetic Expressions Arithmetic Operators an Precedence

All Operators and their Precedence

elational Expressions

Assignment Expressions

Increment, Decrement Operators

Arithmetic if Expressions

Casts

tatements

ple Statements

Scope Looping Statement

while Statemen

Avoid Confusing == with =
Jsing a constant or single varia

as a test condition

while and the null statement

or Statement Comparing while and for

f and switch Statements

Function

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

String

Pointers

Pointers as Function Paramete

rays and Pointe

Strong relationship between arra and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation

Multidimensional Arrays

Command Line Arguments

Structures

Passing Structures to Function

eading and Writing Files

Binary files
Character I/O

Reading a Line at a time: getline()

Guidelines

Style C

Program Design

Some Trings to Rea

Engineers Nick Urbanik

C Programming for

ho am I? Who are

Overview

C++ and iostream library

Continuing our tour

ata Types

xpressions

Arithmetic Expressions Arithmetic Operators and

All Operators and their Precedence

elational Expressions ogical Expressions

Increment, Decrement Operators

Comma Operator
Arithmetic if Expressions

Statemente

Otatomont

Arrays

Arrays

Pointers

Arrays and Pointers

viulitaimensional Arrays and arrays o

Increment and decrement assignment operators

- an assignment operation in which 1 is added/subtracted to a variable and the result assigned to that variable.
- ► The increment or decrement operator can operate in two ways: Prefix places the increment or decrement operator before its operand. Postfix places the increment or decrement operator after its operand.
- Example:

```
x = 32;

y = ++x;

imply x = 32 + 1 = 33 and y = x = 33
```

▶ But

$$\begin{aligned}
 x &= 32; \\
 y &= x++;
 \end{aligned}$$

imply
$$y = x = 32$$
 and $x = 32 + 1 = 33$

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrear brary

Sommung our to

Data Types

Arithmetic Expressions
Arithmetic Operators and

Il Operators and their recedence

neiauonal Expressions Logical Expressions Assignment Expressions

Increment, Decrement Operators

Arithmetic if Expression Bitwise Operators

tatements

.

Arrays

A D.

Arrays and Pointer

Arrays and arrays o

Program plusequl.cpp

```
#include < iostream >
int main( void )
{

    int n, j, k;
    k = j = n = 4;
    std::cout \ll "n = " \ll n \ll ", j = " \ll j
    \ll ", k = " \ll k \ll ' \backslash n';
    n + = j = 3;
    std::cout \ll "n = " \ll n \ll ", j = " \ll j
    \ll ' \backslash n';
}
```

Output for Program plusequl.cpp:

```
n = 4, j = 4, k = 4

n = 7, j = 3
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream library

Continuing our to

ata Types

Arithmetic Expressions

All Operators and their

Relational Expressi

ogical Expressions ssignment Expression

Increment, Decrement Operators

Arithmetic if Expressions
Bitwise Operators
Casts

Statements

Arrays

Pointers

Arrays and Pointers

Arrays and arrays of

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output
Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

Continuing our to

hello.cpp Rasic Synta

Input characters without skipping whitespace

I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output wi

ostream Floating Point

Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

elational Expressions

Assignment Expressions

Comma Operator

Arithmetic if Expressions Bitwise Operators

tatomonte

iple Statements

Scope Looping Statements

while Statement

Avoid Confusing == with =

Jsing a constant or single varial

s a test condition

vhile and the null statement

Comparing while and for

f and switch Statements

Function

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Paramete

rays and Pointe

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Memory Allocation
Multidimensional Arrays

Command Line Argumen

Structures

Passing Structures to Function typedef

ading and Writing Files

Error Handling
Binary files

Reading a Line at a time:

I/O of other data to/from Text F

Guidelines

Program Design

Some Things to Real

C Programming for Engineers

Nick Urbanik

/ho am I? Who are ou?

Overview

C++ and iostream library

Continuing our tour

ata Types

xpressions

Arithmetic Expressions Arithmetic Operators and

All Operators and their Precedence

Logical Expressions
Assignment Expressions

Operators
Comma Operator

Casts

Statemente

Otatomonto

Arrays

Arrays

Pointers

Arrays and Pointers

rrays and arrays of

- The comma operator expression is an expression that consists of a list of other expressions.
- The comma does not perform any operation on these expressions.
 - they are simply evaluated sequentially as if they were a series of statements.
- Example:

$$r = (3 * 5, 8.00 + 2.5, num = 5);$$

▶ The result is the value of the last expression in its list i.e.,

$$r = num = 5$$

► The main use of comma is in the headers of for loops:

for
$$(i = 0, j = n; i < n; ++i, --j)$$

Who am I? Who are You'

Quick Tour

Redirecting Output and Input

C++ and iostream library

An overview of the iostre

An overview of the iostream library

Continuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

Floating Point Types Named Constants

Named Constants
Enumerated Types: enur

Expressions

Arithmetic Expressions Arithmetic Operators an Precedence

All Operators and their Precedence

Relational Expressions
Logical Expressions
Assignment Expressions

Arithmetic if Expressions

Bitwise Operators

atements

ple Statements

Scope Looping Statements

while Statement do statement

void Confusing == with =

Ising a constant or single variable

s a test condition

while and the null statement

Comparing while and for

f and switch Statements

Functions

Calling Functions

Using return Value from F

Arrays

Defining Arrays

Arrays and Loops

Strings

ointers

Pointers as Function Parameter

rrays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Memory Allocation Multidimensional Arrays

Command Line Arguments: a

Structures

Passing Structures to Function typedef

leading and Writing Files

Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text Fil

Guidelines

Program Design Modules

Some Things to Read

C Programming for Engineers

Nick Urbanik

ho am I? Who are u?

Jverview

C++ and iostream library

Continuing our tour

ata Types

pressions

rithmetic Expressions

Arithmetic Operators and Precedence

All Operators and thei Precedence Belational Expression

ogical Expressions
Assignment Expressions

omma Operator

Arithmetic if Expressions
Bitwise Operators

Statomonte

Arrays

Arrays

Pointers

Arrays and Pointers

viulitaimensional Arrays and arrays o

Arithmetic if Expressions

- A conditional expression implements an if-else decision format.
- The conditional expression consists of three subexpressions: test expression and two alternative result expressions (expression1) ? (expression2): (expression3)
- If the test is true, the result will be the value of the second expression.
- If the test is false, the result will be the value of the third expression.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostread

ontinuing our tou

ta Types

Expressions
Arithmetic Expressions

All Operators and the Precedence

Relational Expression
Logical Expressions

Assignment Expressi ncrement, Decreme Operators

Comma Operator

Arithmetic if Expressions

Casts

Statements

Functions

Arrays

Pointers

Arrays and Pointers

lultidimensional rrays and arrays of

Arithmetic if expressions — 2

When used in an assignment operation, the arithmetic if expression works like an if-else statement.

$$max = (a > b) ? a : b;$$

equivalent to

if
$$(a > b)$$
 $max = a$; **else**

max = b;

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

Continuing our tou

ata Types

Expressions

Arithmetic Operators and

Il Operators and their

Relational Expression

ssignment Expression crement, Decrement perators

Arithmetic if Expressions

Casts

Statements

.

Arrays

Arrays and Pointers

Multidimensional Arrays and arrays of

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

library

Continuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard output

and from standard inp

Reading into strin Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

Floating Point Ty

Named Constants Enumerated Types: enum

Expressions

Arithmetic Expressions
Arithmetic Operators and
Precedence

All Operators and their Precedence

Relational Expressions

Logical Expressions
Assignment Expressions
Increment, Decrement Operator

Comma Operator

Bitwise Operators

Casts

ole Statements

Compound Statements

Looping Statement

do statement

woid Confusing == with = Ising a constant or single variab as a test condition

while and the null statement

Comparing while and for

f and switch Statements

Function

Calling Functions

Using **return** Value from Fu

Arrays

Defining Arrays

Arrays and Loops

String:

Pointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Memory Allocation
Multidimensional Arrays

Command Line Argumen

Structures

Passing Structures to Function typedef

ading and Writing Files

Error Handling Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text Fil

Guideline

Program Design

Some Trings to Hea

Engineers Nick Urbanik

C Programming for

/ho am I? Who are ou?

Jverview

C++ and iostrear library

ontinuing our tour

ata Types

pressions

ithmetic Expressions

Arithmetic Operators and Precedence

All Operators and theil Precedence Relational Expression

ogical Expressions

Operators Comma Operator

Arithmetic if Express
Bitwise Operators

Otatomonto

Arrays

Arrays

Pointers

Arrays and Pointer

viulitaimensional Arrays and arrays o

Bitwise operators — 1

- ► The bitwise operations allow the programmer to manipulate specific bits.
- ► The bitwise operations can be combined with *masks* to turn specific bits on and off.
- ➤ The bitwise AND operation, &, is used to clear specific bits in an integer operand, leaving the other bits unchanged.
- ► The bitwise *OR* operation, |, is used to *set* specific bits in an integer operand, leaving the other bits unchanged

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C<mark>++ and</mark> iostrea brary

onlinuing our t

ata Types

Expressions
Arithmetic Expressions
Arithmetic Operators and
Precedence

Antimetic Operators and Precedence

All Operators and their Precedence

Logical Expressions
Assignment Expressions
Increment, Decrement
Operators

Comma Operator Arithmetic if Expressi

Bitwise Operators Casts

Statements

Arrave

Arrays

Arrays and Pointer

Multidimensional Arrays and arrays o

Bitwise operators — 2

▶ Here is the truth table for the bitwise AND operation:

bit n		bit m		m	& n
Т	1	Т	1	Т	1
Т	1	F	0	F	0
F	0	Т	1	F	0
F	0	F	0	F	0

Truth table for the bitwise OR operation:

bit n		bit m		m	l n
Т	1	Т	1	Т	1
Т	1	F	0	Т	1
F	0	Т	1	Т	1
F	0	F	0	F	0

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea brary

Continuing ou

ata Types

Express

Arithmetic Expre

thmetic Operators and

I Operators and th recedence

Relational Expression

Assignment Expressions

Operators Comma Operator

Arithmetic if Expres

Bitwise Operators

Otatomont

Arrays

Allays

Arrays and Point

Multidimensional Arrays and arrays of

Bitwise operators — 3

- ▶ The one's complement operator, \sim , is a unary operator.
- ▶ The resulting value is set to the opposite of that of the operand's bit.

bit m	~m
0	1
1	0

The bitwise exclusive OR operator, ^, results in 1 if the corresponding two bits are different:

bit m	bit m	m ^ n
0	0	0
0	1	1
1	0	1
1	1	0

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

++ and iostr rary

continuing our

ata Types

Arithmetic Expressions

Operators and their recedence

Relational Expression Logical Expressions

ocrement, Decrement Operators Comma Operator

Arithmetic if Express
Bitwise Operators

Statements

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Shift Operators

- There are two shift operators:
 - left shift operator
 - >> right shift operator
- useful for accessing individual parts of a bit pattern
- shift the bits of the left operand some number of positions to the left or right.

```
      unsigned char bits = 1;
      // 0000 0001

      bits = bits \ll 1;
      // 0000 0010

      bits = bits \ll 2;
      // 0000 1000

      bits = bits \gg 3;
      // 0000 0001
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our to

ata Types

Expressio

Arithmetic Expressions
Arithmetic Operators and

Il Operators and the recedence

Relational Expression:

ogical Expressions

ncrement, Decrement Operators

Comma Operator Arithmetic if Expre

Bitwise Operators

Statements

Otatomonto

.

Arrays

Arrays and Pointers

Multidimensional Arrays and arrays o

- the right shift operator will fill negative signed numbers with '1's from the left, but will shift 0 into the MSb (most significant bit) of unsigned numbers.
- Program shiftright-demo.cpp

```
#include < iostream>
#include < iomanip>
int main( void )
        int test_numbers[] = \{ -16, 16, -1, 1 \}:
        const int len = sizeof( test_numbers ) / sizeof( test_numbers[ 0 ] );
        for ( int i = 0; i < len; ++i ) {
                int n = test_numbers[i]:
                unsigned u = n;
                std::cout << showbase:
                std::cout \ll "dec n: " \ll dec \ll n \ll "; n \gg 2: " \ll (n \gg 2\sim)hmetic if Expressions
                     \ll "\thex n: " \ll hex \ll n \ll "; n \gg 2: " \ll ( n \gg 2 )
                     << '\n':
                std::cout \ll "dec u: " \ll dec \ll u \ll "; u \gg 2: " \ll (u \gg 2) ements
                     \ll "\thex u: " \ll hex \ll u \ll "; u \gg 2: " \ll ( u \gg 2)
                     << '\n':
```

Bitwise Operators

Right Shift Operator: example output

▶ output of program shiftright-demo.cpp (folded to fit):

- Notice that when shifted right:
 - unsigned values have the upper two bits zero
 - ▶ so the unsigned value 0xffffff0 shifted right by two is 0x3ffffffc
 - signed values have the upper two bits the same as the previous value of the MSb (most significant bit)
 - so the signed value 0xfffffff0 (-16) shifted right two places is 0xffffffc

C Programming for Engineers

Nick Urbanik

/ho am I? Who are

Overview

C++ and iostream ibrary

Continuing our

ata Types

Expressions

Arithmetic Operators and

Precedence

ogical Expressions

ssignment Expression crement, Decrement perators

Comma Operator Arithmetic if Expres Bitwise Operators

Casts

Statements

Tunction

Arrays

Pointers

Arrays and Pointe

Multidimensional Arrays and arrays

Use of bitwise operators

- Use the AND operator '&' to clear individual bits, leaving the others unchanged
 - ▶ For example,

$$x = x \& 0xf;$$

will clear all but the least significant four bits of x

- ▶ Use the OR operator '|' to set individual bits, leaving the others unchanged
 - ► For example,

$$y = y \mid \tilde{o}xf;$$

will set all bits except for the least significant four bits of y

- ▶ Use the Exlusive OR operator '^' to toggle (flip) individual bits, leaving the others unchanged
 - For example,

$$z = z ^ 0xf;$$

will toggle the least significant four bits of z, i.e., make $1s \leftrightarrow 0s$

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrear brary

ontinuing our tou

ata Types

rithmetic Expressions

rithmetic Operators and recedence
II Operators and their

Relational Expression
ogical Expressions

crement, Decrement perators omma Operator

Arithmetic if Expres
Bitwise Operators

Casts

Statements

.

Arrays

Pointers

Arrays and Pointer

ultidimensional rays and arrays of

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

An overview of the iostream library

itinuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard outp

and from standard inp

Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

Floating Point Ty

Named Constants Enumerated Types: enur

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions

Assignment Expressions
Increment, Decrement Operato

Arithmetic if Expressions

Casts

tements

ple Statements

Scope _ooping Statements

while Statement

Avoid Confusing == with =

Jsing a constant or single varial

Jsing a test condition

while and the null statement

or Statement Comparing while and for

f and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Fu

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Memory Allocation

Multidimensional Arrays

Command Line Arguments:

Structures

Passing Structures to Function typedef

eading and Writing Files

Error Handling Binary files

Reading a Line at a time:

I/O of other data to/from Text Fil

auldelines

Program Design Modules

Some Things to Rea

Engineers Nick Urbanik

C Programming for

ho am I? Who are

Jverview

C++ and iostream

Continuing our tour

ata Types

xpressions

Arithmetic Expressions
Arithmetic Operators and

Precedence
All Operators and their

Precedence
Relational Expressions

ogical Expressions
Assignment Expressions

Comma Operator Arithmetic if Express

Bitwise Operators

01-1----

Arrays

Allays

Pointers

Arrays and Pointer

rrays and arrays of

Cast expressions

- The cast operation returns the value of an expression, converting it to the type in the brackets
- Example:

```
( float ) 7
```

- ► This converts the integer 7 to a floating point value, 7.00.
- The operand may be any expression
 - Examples:

```
( int ) ( 5.8 * 2.7 )
( double ) ( k = 10 )
( float ) num
```

- If a floating point value is cast to an integer, the floating point fraction is lost.
 - Example:(int) 3.75
 - resulting value is 3
 - Casts overide the compiler's concept of correctness use rarely

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

G++ and iostre brary

onunuing our i

ata Types

xpressions
Arithmetic Expressions
Arithmetic Operators and

All Operators and the Precedence
Relational Expression

Logical Expressions
Assignment Expressions
Increment, Decrement
Operators
Comma Operator

Arithmetic if Express
Bitwise Operators
Casts

. .

Statements

Functions

Arrays

_ .

Arrays and Pointe

Multidimensional Arrays and arrays

Conversions

- Expressions can include operands of different number types
- Example: An integer can be multiplied by a float.
- C handled operands of different types by converting one of them into the same type as that of the other operand.
- Conversion determines which operand to convert by a process of promotion and demotion.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and ios brary

Continuing our tou

ata Types

expressions

Arithmetic Expressions
Arithmetic Operators and

All Operators and their Precedence

Relational Expressions

Logical Expressions

Assignment Expression Increment, Decrement Operators

> Comma Operator withmetic if Express

Arithmetic if Express Bitwise Operators

atements

Caete

Arrays

Pointers

Arrays and Pointers

lultidimensional rrays and arrays (

Conversions: promotion

- ▶ In expressions with two different types of operands, the operand with the smaller type is always *promoted* to that of the largest type.
- Example:

```
int num = 6;
float cost;
cost = num * 5;
```

► The expression *num* * 5 results in the integer 30 and will be promoted to a **float**, 30.0 before assigned into the *cost* variable.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrear brary

Continuing our tou

ata Types

xpressions

Arithmetic Expressions
Arithmetic Operators and

All Operators and th Precedence

Logical Expressions
Assignment Express

Assignment Expressions Increment, Decrement Operators Comma Operator

Arithmetic if Expre Bitwise Operators

tatements

Arrays

Caete

Allays

Arrays and Pointers

fultidimensional rrays and arrays o

Conversions: demotion

- ► *Demotion* from a floating point type to an integer type results in the loss of the floating point's fraction.
- Example:

```
int num;
float cost = 62.65;
num = cost;
```

► The fraction will be cut off, leaving just the integer, 62. The value 62 is then assigned to the variable *num*.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

)verview

C++ and io brary

Continuing our to

ata Types

Arithmetic Expressions

Arithmetic Operators and

All Operators and the Precedence

Relational Expression
Logical Expressions
Assignment Expression

Operators
Comma Operator
urithmetic if Express

Casts

tatements

Functions

Arrays

Pointers

Arrays and Pointers

rrays and arrays (

Statements: an introduction

- A C program is a sequence of declarations and statements.
- We have seen:
 - how to declare variables, and
 - how to create expressions using operators
 - examples of putting this all together.
- Now let's look at statements.

C Programming for Engineers

Nick Urbanik

Statements

do statement

while and the null

for Statement

Statements: intro — 2

We can turn an expression such as x = 0 or std::cout ≪ "a" into a statement simply by putting a semicolon at the end:

```
x = 0;

std::cout \ll "a";
```

- We can join these into a compound statement by putting braces { } around them.
- ► There are some which are used to create loops and make decisions. These are sometimes called control-flow statements.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ontinuing our toi

. .

Expression

Statements

Simple Statements

Compound Statements Scope

ooping Statements

do statement

void Confusin

Using a constant or single variable as a test condition while and the null

tatement

for Statement Comparing will

Comparing while and f if and switch Statemer break, continue, goto

unctions

Pointers

Arrays and Pointers

Who am I? Who are You Overview

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

An overview of the iostream library

ntinuing our to

hello.cpp Basic Synta

Input characters without skipping whitespace

I/O of other data to standard output

and from standard inp Reading into strings

Loops

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output wit

ostream Floating Poir

Named Constants

xpressions

Arithmetic Expressions Arithmetic Operators and

All Operators and their Precedence

lelational Expressions

Assignment Expressions

Increment Degrament Operator

Increment, Decrement Operator

Arithmetic if Expressions
Bitwise Operators

Statements

Simple Statements

imple Statements compound Statement

Scope
Looping Statement

do statement

Avoid Confusing == with =

Using a constant or single varia

as a test condition

while and the null statement

for Statement Comparing while and for

Comparing while and for f and switch Statements

Functions

Defining Functions
Calling Functions

Jsing return Value from F

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

fultidimensional Arrays and arrays o

pointers

Memory Allocation

Multidimensional Arrays

Command Line Argume

Structures

Passing Structures to Function

eading and Writing File

fstream: file input and out

Binary files Character I/O

Reading a Line at a time:

uidelines

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

/ho am I? Who are ou?

Jverview

C++ and iostream library

Continuing our tour

ata Types

pressions

Simple Statements

Compound Statement

Looping Statements

while Statement

Using a constant or variable as a test co

while and the null statement

for Statement

Comparing while and for if and switch Statements break, continue, goto

unctions

unctions

_ .

Pointers

Arrays and Point

Expression statements

- An expression statement consists of any valid expression, followed by a semicolon.
- Often the expression is an assignment operation or a function call.
- Example:

```
count = 8;
num = 3 + 4;
calc();
```

- However, the expression could just as easily be an arithmetic expression or relational expression.
- Example:

```
4 + 5; // nothing done with the result, 9. ( n < 3 );
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream
brary

ontinuing our to

ata Typos

Expressions

Simple Statements

Scope

while Statement

do statement

Avoid Confusing

Using a constant or sin variable as a test condiwhile and the null

tatement

for Statement

if and switch Statement break, continue, goto

unctions

...., -

Pointers

Arrays and Pointers

Null statement (empty statement)

- ▶ If there is no expression in the expression statement, nothing happens. This is called as the *null statement*.
- ► Example:
 - ; // just a semicolon

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

Continuing our tour

ıta Types

Expressions

latements

Simple Statements

Scope Scope

Looping Statements

do statement

do statement

void Confusing == with = sing a constant or single uriable as a test condition

while and the null statement

for Statement

Comparing while and to if and switch Statement break, continue, goto

Eunctions

Functions

Pointers

Arrays and Pointers

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

library

hello.cpp

Input characters without skipping whitespace

I/O of other data to standard outp

Reading into strings

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output wi

Floating Point

Enumerated Types: enur

ynressions

Arithmetic Expressions Arithmetic Operators and All Operators and their Precedence Relational Expressions

Relational Expressions ogical Expressions

Assignment Expressions
Increment, Decrement Operator

Comma Operator
Arithmetic if Expressions
Bitwise Operators

Statements

le Statements

Compound Statements

Looping Statements
while Statement

do statement Avoid Confusing == with = Using a constant or single variab

as a test condition

while and the null statement

for Statement Comparing while and for

if and switch Statements

Functions

Defining Functions
Calling Functions

Using **return** Value from Fi

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers as Function Paramete

rays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays or

pointers

Memory Allocation Multidimensional Arrays

Command Line Arguments: argo

Structures

Passing Structures to Functions
typedef

eading and Writing Files

fstream: file input and outp

Binary files
Character I/O
Reading a Line:

Reading a Line at a time: getline()

Guidelines

Style Guidelines

Modules

Some Things to Real

C Programming for Engineers

Nick Urbanik

ho am I? Who are

Overview

C++ and iostream library

ontinuing our tour

ata Types

.....

nple Statements

Compound Statements

Scope Looping Statements

while Statement

Avoid Confusing == w
Using a constant or si
variable as a test cond
while and the null

while and the null statement

for Statement
Comparing while and for

if and switch Statement break, continue, goto

unctions

riidys

Pointers

Arrays and Pointe

Compound Statements

- A compound statement is a statement composed of one or more statements.
- ► A compound statement consists of opening and closing braces within which statements are placed.
- Example:

```
{  num = 6; \\ fact = (5-3); \\ std::cout \ll fact \ll ", " \ll num \ll ' \n'; \\ \}
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our to

ata Types

Expression

tatements

Compound Statements

Scope

Looping Statements

do statement

Avoid Confusing

Using a constant or single variable as a test condition while and the null

tatement

for Statement

if and switch Statemer break, continue, goto

unctions

Pointers

Arrays and Pointers

Blocks — 1

- Variables can be declared at the beginning of compound statement. A compound statement with variable declarations is referred to as a block.
- ► The body of a function is a compound statement itself, and is often referred to as the *function block*.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream
library

ontinuing our tour

ata Types

Expressions

atements

Compound Statements

Scope Looping Statements

hile Statement

do statement

oid Confusing == with = ing a constant or single riable as a test conditio

while and the null statement

for Statement

Comparing while and fif and switch Statement break, continue, goto

unctions

Functions

Pointore

Assessed Delies

Program blocks.cpp

```
#include < iostream>
int main( void )
       int num = 10;
       float cost = 100.0;
               float cost = 50.0:
               num = 5:
               std::cout ≪ "Inside: " ≪ cost
                    \ll ", " \ll num \ll '\n';
       std::cout << "Out.side: " << cost
            \ll ", " \ll num \ll '\n':
```

▶ Output of program blocks.cpp:

```
Inside: 50, 5
Outside: 100, 5
```

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostre brary

ontinuing our tour

ata Types

Expressions

atements

Compound Statements

Scope

while Statement

do statement Avoid Confusing

Using a constant or single variable as a test condition while and the null statement

break, continue, goto

for Statement

FULLCUOIS

_ .

Pointers

rrays and Point

slide 122/256

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

library

Continuing our to

hello.cpp Rasic Synta

Input characters without skipping

I/O of other data to standard outp

and from standard inpu Reading into strings

Reading into str

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output wit

ostream Floating Poi

Named Constants

ynressions

Arithmetic Expressions
Arithmetic Operators and

All Operators and their Precedence

lelational Expressions

Assignment Expressions
Increment Decrement Operator

Increment, Decrement Operator
Comma Operator

Bitwise Operators

Statements

Simple Statements

Compound Sta Scope

Looping Statements while Statement

do statement
Avoid Confusing == with =
Using a constant or single variables a test condition

while and the null statement

Comparing while and for

f and switch Statements

Functions

Defining Functions
Calling Functions

Jsing **return** Value from Fi

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

ays and Pointers

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Arguments

Structures

Passing Structures to Functions

typedef

ading and Writing Files

Error Handling Binary files Character I/O

Reading a Line at a time:

i/O of other data to/from Text F Guidelines

Style Guidelines Program Design

Modules

Some Things to Read

C Programming for Engineers

Nick Urbanik

ho am I? Who are

Overview

C++ and iostream library

ontinuing our tour

ata Types

xpressions

atements

imple Statements ompound Statements

Scope

while Statement

do statement

Using a constant or si variable as a test con-

while and the null statement

for Statement
Comparing while and for

if and switch Statemer break, continue, goto

unctions

ranays

Pointers

Arrays and Poin

Scope

A new scope is created in each block

- the compiler searches for an identifier defined in the innermost scope first
- then searches the scopes that the enclose current scope...
- ... until it reaches global scope
 - global scope is outside of any block
- identifiers defined in an inner scope "hide" identifiers defined in an outer scope
- ► For example, in program blocks.cpp, there are two variables called *cost* in different nested scopes
 - The inner output statement prints the value of the cost variable defined in the inner scope
 - The outer output statement prints the cost defined in the outer scope.

C Programming for Engineers

Nick Urbanik

Scope

do statement

while and the null

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

An overview of the iostream library

ontinuing our to

hello.cpp Basic Synta

Input characters without skipping whitespace

I/O of other data to standard outp

and from standard inp Reading into strings

Loops

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output with

ostream Floating Poi

Named Constants

xpressions

Arithmetic Expressions Arithmetic Operators and All Operators and their Precedence

Relational Expression ogical Expressions

Assignment Expressions Increment, Decrement Operator

Comma Operator
Arithmetic if Expressions
Bitwise Operators

Bitwise Operatoi Casts

Statements

ole Statements

Looping Statements

while Stateme

do statement Avoid Confusing == with = Using a constant or single variab as a test condition

while and the null statement

Comparing while and for

f and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Fu

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

rays and Pointer

Strong relationship between array and pointers

lultidimensional Arrays and arrays of

pointers

Memory Allocation Multidimensional Arrays

Command Line Argume

Structures

Passing Structures to Function

eading and Writing Files

Error Handling Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text Fi

Style Guideline

Modules

Some Things to Rea

C Programming for Engineers

Nick Urbanik

/ho am I? Who are

Overview

C++ and iostream library

ontinuing our tour

ata Types

xpressions

itements

Simple Statements Compound Statements

Looping Statements

while Statement

do statement Avoid Confusing

Using a constant or s variable as a test cor while and the null

while and the null statement

for Statement

if and switch Statement break, continue, goto

unctions

Arrays

Pointers

Arrays and Pointe

Iteration statements

- Loops are implemented with three iteration statements:
 - while
 - do
 - ▶ for
- Each statement repeats a statement called the body of the loop
- ► The body of the loop can be any statement, including:
 - compound statement, or
 - another loop, or
 - a null statement.

C Programming for Engineers

Nick Urbanik

Looping Statements

do statement

while and the null

for Statement

All Operators and their Precedence

Statements

while Statement

Arrays and Loops

C Programming for Engineers Nick Urbanik

while Statement

do statement

while and the null

Comparing while and for

while statement

- while loop consists of the keyword while, a test expression in parentheses, and a statement.
- statement is repeated for as long as the test expression evaluates to a non-zero value.

```
while ( \langle test expression \rangle )
              (statement);
```

Example:

```
i = 0; // initialize the loop counter
while ( i < 5 ) {
   std::cout ≪ "ABC ";
    ++i; // update the loop counter
         // very important
         // the loop becomes infinite
         // without this statement that
         // changes the loop counter
```

C Programming for Engineers

Nick Urbanik

while Statement do statement

while and the null

for Statement

All Operators and their Precedence

Statements

do statement

Arrays and Loops

C Programming for Engineers

Nick Urbanik

while Statement

do statement

while and the null

Comparing while and for

Iteration Statements — do loop

- ▶ The **do** statement is a variation on the while statement.
- ► Instead of the test occurring at the beginning of the loop, it occurs at the end,
- Example:

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

O<mark>++ and</mark> iostream ibrary

Continuing our tou

ata Types

Expressions

mple Statements

Scope

poping Statements

hile Statemer

do statement

Avoid Confusing == with =
Using a constant or single
variable as a test condition
while and the null

statement

for Statement

if and switch Statemer break, continue, goto

Functions

Pointers

Arrays and Pointers

C++ and iostream

.... ..

- .

Expressions

nple Statements mpound Statements

Scope Looping Statements

while Statement

do statement

Avoid Confusing == with = Using a constant or single variable as a test condition while and the null

for Statement

or Statement Comparing wh

break, continue, goto

Functions

Pointers

rays and Poir

```
#include < iostream>
int main( void )
       int num;
       std::cout ≪ "Enter a number: ";
       std::cin >> num:
       while ( num != 0 ) {
               int square = num * num;
               std::cout ≪ "Square of " ≪ num
                    \ll " = " \ll square \ll '\n';
               std::cout << "Enter a number (0 to quit): ";</pre>
               std::cin >> num:
```

- ► This program has a major problem.
- What happens if we input a character that is not part of an integer?

All Operators and their Precedence

Statements

Avoid Confusing == with =

Arrays and Loops

C Programming for Engineers Nick Urbanik

do statement

Avoid Confusing == with =

while and the null

Comparing while and for break, continue, goto

Test Expression

- ► The test expression for the while, for, and if statements can be any valid expression,
- Example:
 - assignment operation
 - simple primary expression consisting of a variable or a constant
- A zero result evaluated is considered to be false while any non-zero result is true.

C Programming for Engineers

Nick Urbanik

do statement

Avoid Confusing == with =

while and the null

for Statement

```
while (i = k) \{ ... \}
```

- ▶ If *k* equals 0, the test will always be false
- ightharpoonup If k is not equal to zero, the test will always be true.
- ► Examples of *incorrect* test expressions:

```
while ( n=0 ) { /* always false */ ... } while ( n=3 ) { /* always true, an infinite loop */ ... }
```

The correct way to write these test expressions is:

Who am I? Who are You?

Overview

++ and iostream orary

ontinuing our tou

ata Types

Expressions

Simple Statements

cope

ooping Statements

do statement

Avoid Confusing == with =

variable as a test cowhile and the null statement

Comparing while and f if and switch Statemen break, continue, goto

Functions

Arrays

Pointers

Arrays and Pointers

Program quit-1.cpp

```
#include < iostream>
/* This program shows problems that come from
   confusing assignment with comparison */
int main( void )
        int quit = 0, num = 1, square;
        while ( quit = 0 ) // Oh dear; always false
                // the loop body will never be executed
                square = num * num;
                std::cout ≪ "Square of " ≪ num
                     \ll " = " \ll square \ll '\n';
                if ( num = 10 ) // Oh dear; always true
                        quit = 1;
                num++:
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

C++ and iostream

Continuing our tou

ata Types

Expressions

Compound Statement Scope

ooping Statements

do statement

Avoid Confusing == with =

Ising a constant or sing ariable as a test condit

while and the null statement for Statement

Comparing while and to if and switch Statement break, continue, goto

Functions

Pointers

Arrays and Pointers

Who am I? Who are You?

Quick Tour

Standard Input, Standard Output
Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ontinuing our to

hello.cpp Basic Synta

Input characters without skipping whitespace

I/O of other data to standard outp

and from standard inp

Reading into strings
Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output wit

Floating Point Ty

Enumerated Types: enum

xpressions

Arithmetic Expressions Arithmetic Operators and

All Operators and their Precedence

Relational Expressions

Assignment Expressions

Increment, Decrement Operator

Arithmetic if Expressions Bitwise Operators

Statements

ple Statements

Scope Looping Statement

while Statement

void Confusing == with

Using a constant or single variable as a test condition

while and the null statement

Comparing while and for

if and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Funct
Function Parameters

Arrays

Defining Arrays

Arrays and Loops

Strings

Pointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays of

pointers

Memory Allocation
Multidimensional Arrays

Command Line Argumer

Structures

Passing Structures to Function

eading and Writing File

fstream: file input and output Error Handling

Character I/O
Reading a Line at a time

getline()
I/O of other data to/from Text Fil

Guidelines

Program Design

some Things to Rea

C Programming for Engineers

Nick Urbanik

no am I? Who are

Overview

C++ and iostream library

ontinuing our tour

ata Types

xpressions

ements

Simple Statements
Compound Statements

cope coping Statements

while Statement do statement

Avoid Confusing == with =

Using a constant or single variable as a test condition while and the null

while and the null statement

Comparing while
if and switch Stat

break, continue, goto

unctions

_ .

Pointers

arayo and rom

Using a constant or single variable as a test condition

- ► Constants are often used to write infinite loops.
- Variables are used as a shorthand for comparing the value of the variable to zero.
- Example:

C Programming for Engineers

Nick Urbanik

wno am 17 wno are You?

Overview

C++ and iostream brary

onlinuing our to

ata Types

Expressions

tatements

nple Statements

Scope

ooping Statements

do statement

o statement

Using a constant or single

variable as a test condition while and the null

while and the null statement

for Statement

if and switch Statemer break, continue, goto

Functions

Pointers

rrave and Point

Iultidimensional

Program square-2.cpp

```
#include <iostream>
int main( void )
       int num;
       while (1) { // or for (;;)
               std::cout ≪ "Please enter a number: ";
               if (! ( std::cin >>> num ) )
                       break:
               int square = num * num;
               std::cout ≪ "Square of " ≪ num
                    \ll " = " \ll square \ll '\n';
               char ch;
               std::cout \ll  "Another square? (y/n) ";
               if ( std::cin \gg ch \&\& ch != 'y' )
                       break:
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

ontinuing our tou

ata Types

Expressions

ple Statements

Scope

ooping Statements

do statement

Avoid Confusing

Using a constant or single

variable as a test condition while and the null statement

statement for Statement

Comparing while

break, continue, goto

Functions

Pointers

Arrays and Pointers

All Operators and their Precedence

Statements

while and the null statement

Arrays and Loops

C Programming for Engineers Nick Urbanik

while Statement do statement

while and the null statement

break, continue, goto

while test and null statement

➤ A while statement can be written in which the test expression does all the work. The statement following the test expression is simply a null statement.

Example:

```
while ( std::cin.get( ch ) && ch != '\n' ) ; // do nothing
```

- ▶ Three actions take place in the above example:
 - function call to the istream member function get()
 - std::cin.get(ch) will return false if reach end of input file;
 - inequality operation in which character value obtained by std::cin.get() is tested against a newline constant.
- End result is that we wait till we get a newline character, or end of file

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostre brary

Johnnang our

ata Types

Expressions

tatements

Compound Statement

Looping Statements

hile Statement

do statement

oid Confusing == with = sing a constant or single riable as a test condition

while and the null statement

or Statement

Comparing while and for if and switch Statements break, continue, goto

unctions

rravs

Pointers

Arrays and Pointe

Multidimensional

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output
Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

Continuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard outp

and from standard inp

Reading i Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output with

Floating Point Types
Named Constants

Enumerated Types: enum

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions
Assignment Expressions

Increment, Decrement Operator

Arithmetic if Expressions
Bitwise Operators

Statements

ole Statements

Scope Looping Statements

while Statement do statement

Avoid Confusing == with =
Using a constant or single variations a test condition

for Statement

Comparing while and for if and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Fund

Arrays

Defining Arrays

Arrays and Loops

Strings

ointers

ays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation Multidimensional Arrays

Command Line Arg

Structures

Passing Structures to Function typedef

ng and Writing File

fstream: file input and outp Error Handling

Binary files Character I/O

Reading a Line at a time: getline()

Guidelines

Style Guide

Program Design

Some Things to Rea

C Programming for Engineers

Nick Urbanik

ho am I? Who are u?

Overview

C++ and iostream library

ontinuing our tour

ata Types

xpressions

tements

Simple Statements Compound Statements

cope coping Statements

while Statement do statement

void Confusing

Using a constant or sin variable as a test cond while and the null

for Statement

Comparing while and for if and switch Statements break, continue, goto

unctions

_ .

Pointers

Arrays and Point

The **for** Statement

► The for statement consists of three expressions followed by a statement.

```
for ( \langle expression 1 \rangle; \langle expression 2 \rangle; \langle expression 3 \rangle) \langle statement \rangle;
```

- \(\expression 1\)\(\text{ is executed once, before loop begins. It is often used to \(\frac{initialize}{initialize}\)\(\text{ variables used in the test expression.}\)
- \(\end{expression 2}\)\) is the test expression for the loop. When it evaluates as false, the loop stops.
- \(\end{array}\) \(\text{update}\) expression. It is executed within the loop and is usually used to update variables used in the test expression.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

++ and iostream orary

ontinuing our tou

oata types

Expressions

Simple Statements
Compound Statement

cope coping Statements

hile Statement

do statement

Avoid Confusing

Using a constant or sing variable as a test condit while and the null statement

for Statement

Comparing while and for if and switch Statements break, continue, goto

unctions

Pointers

rrays and Pointe

slide 142/256

ultidimensional

Example of for loop

```
for ( int i = 1; i < 3; ++i ) { std::cout \ll "OK\n"; }
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream library

Continuing our tour

Data Types

Expression

tatements

Simple Statements Compound Statements

cope poping Statements

Looping Statements
while Statement

do statement

Avoid Confusing == with = Using a constant or single variable as a test condition while and the null

for Statement

Comparing while and for if and switch Statements break, continue, goto

Functions

_ .

Pointers

Arrays and Pointers

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ntinuing our to

hello.cpp Basic Synta

Input characters without skipping whitespace

I/O of other data to standard outp

and from standard inp

Reading into strir Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output wi

ostream Floating Poir

Named Constants
Enumerated Types: enum

ynressions

Arithmetic Expressions Arithmetic Operators and

All Operators and their Precedence

Relational Expressions

Logical Expressions
Assignment Expressions

Increment, Decrement Operator

Arithmetic if Expressions
Bitwise Operators

Statements

ole Statements

Scope Coping Statements

while Statemer

do statement Avoid Confusing == with = Using a constant or single varial as a test condition

or Statement

Comparing while and for

and **switch** Statement eak, continue, goto

unctions

Defining Functions

Using return Value from Function Parameters

Arrays

Defining Arrays

Arrays and Loops

Strings

ointers

Pointers as Function Paramete

ays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays of

pointers

Memory Allocation Multidimensional Arrays

Command Line Argume

Structures

Passing Structures to Function

ading and Writing File

fstream: file input and outp Error Handling

Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text F

duidelines

Program Design

Some Things to Rea

C Programming for Engineers Nick Urbanik

Quarriou

C++ and iostream

ontinuing our tour

ata Types

kpressions

tements

Simple Statements
Compound Statements

cope

while Statement

do statement

Using a constant or sin variable as a test cond while and the null

while and the null statement

Comparing while and for

if and switch Statement break, continue, goto

Functions

Arrays

Pointers

Arrays and Poin

for and while: a comparison

```
for loop:
                                                      while loop:
for ( \(\langle init \rangle ; \(\langle test \rangle ; \(\langle update \rangle \)) \{
                                                      ⟨init⟩;
                                                      while ( \langle test \rangle ) {
             (body of loop)
                                                                   ⟨body of loop⟩
                                                                   (update):
example:
                                                      example:
for ( int i = 0; i < 3; ++i)
                                                      int i = 0;
          std::cout << "loop" << i
                                                      while (i < 3)
                      << '\n':
                                                                std::cout << "loop" << i
                                                                            << '\n':
                                                                ++i:
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

Continuing our tou

ata Types

Expressions

Simple Statements

Compound Statement

oping Statements

while Statement

do statement

Using a constant or singly variable as a test condition while and the null statement for Statement

Comparing while and for

break, continue, goto

Functions

Pointers

Arrays and Point

Aultidimensional

Nested loops — loop within a loop

- The inner loop executes fully within each iteration of the outer loop
- Example:

```
for ( int k = 0, i = 0; i < 3; ++i ) {
    for ( int j = 0; j < 3; ++j ) {
        ++k;
        std::cout \ll k \ll ' ';
    }
    std::cout \ll ' \n';
}
```

- See example program nested-for.cpp
- Output:

```
1 2 3
4 5 6
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our tol

ta Types

Expressions

ntements

Simple Statements

Scope

Looping Statements
while Statement

do statement

o statement

Using a constant or single variable as a test condition while and the null

while and the null statement for Statement

for Statemen

Comparing while and for if and switch Statements break, continue, goto

Functions

Arrays

Pointers

Arrays and Pointer

Who am I? Who are You Overview

Quick Tour

Standard Input, Standard Output
Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

tinuing our to

hello.cpp Basic Synta

Input characters without skipping whitespace

I/O of other data to standard outp

and from standard in Reading into strings

Loops

Reading till the end of file

Data Types

Integer Types Characters

Octal, Hexadecimal Output w

Floating Poi

Enumerated Types: enum

xpressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions

Assignment Expressions
Increment, Decrement Operator

Comma Operator

Arithmetic if Expressions Bitwise Operators

Statements

ole Statements

Scope

while Statemer

do statement
Avoid Confusing == with =
Using a constant or single varia

while and the null statement

Comparing while and for

if and switch Statements

break, continue, goto

unctions

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

String

Dointore

Pointers as Function Parameter

ays and Pointe

Strong relationship between array and pointers

lultidimensional Arrays and arrays o

pointers

Memory Allocation

Multidimensional Array

Command Line Argument

Structure

Passing Structures to Function vpedef

iding and Writing File

fstream: file input and outp

Binary files Character I/O

Reading a Line at a time:

I/O of other data to/from Text Fil

iuidelines

Program Design Modules

Some Things to Rea

C Programming for Engineers

Nick Urbanik

ho am I? Who are

Dverview

U++ and lostrea: library

ontinuing our tour

ata Types

nraccione

tomonte

mple Statements

соре

while Statement

do statement

Using a constant or s variable as a test con

while and the null statement

for Statement
Comparing while and for

if and switch Statements break, continue, goto

unctions

Pointers

Arrays and Point

Conditions: if statement

- ► The if and switch statements are decision making structures that determine which statements are to be executed and which are not.
- ▶ if is a condition placed on a statement's execution.
 - if the condition is true, the statement is executed
 - if the condition is false, the statement is not executed

```
if ( \langle test expression \rangle )
  \langle statement \rangle;
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream
library

ontinuing our tou

ata Types

Expressio

Statements

Compound Stateme

Scope Looping Statements

ooping Statements

do statement

statement

oid Confusing == with ng a constant or singliable as a test condi-

while and the null statement

or Statement

Comparing while and for if and switch Statements

break, continue, goto

-unctions

Arrays

Pointers

rrays and Pointe

lultidimensional

if and else

► The if statement can choose between several choices using else:

- ► Only one action is performed
- the first that matches is done.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream

Continuing our tour

ata Types

Expression

imple Statements

Compound Statemen

Looping Statements

while Statement

do statement

do statement

Using a constant or sing variable as a test conditions while and the null

tatement

for Statement

Comparing while and for if and switch Statements

break, continue, goto

Functions

.

Pointers

Arrays and Pointe

The **switch** statement — 1

- The switch statement provides a convenient way to choose among several alternatives.
- It is a conditional statement (selection)

```
switch ( \( \langle integer expression \rangle \) \( \{ \)
      case (integer):
            ⟨statements⟩;
            break:
      case (integer):
            (statements);
            break;
      default:
             (statements);
            break;
```

C Programming for Engineers

Nick Urbanik

do statement

while and the null

for Statement

if and switch Statements

break, continue, goto

The switch statement — 2

- ► The **switch** compares an *integer value* against a set of integer constants.
- ► The execution will continue unless a **break** is encountered.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

Continuing our tou

pata Types

Expressions

. . . .

imple Statements

Compound Statements

Looping Statements

Looping Statements

do statement

lo statement

oid Confusing == with ng a constant or single able as a test condition

while and the null statement

for Statement

Comparing while and for if and switch Statements break, continue, goto

Functions

Turicuoris

Pointers

Arrays and Pointers

All Operators and their Precedence

Statements

break, continue, goto

Arrays and Loops

C Programming for Engineers

Nick Urbanik

while Statement do statement

while and the null

Comparing while and for

break, continue, goto

The jump Statements

- The jump statements are non-structured control statements that allow the program to jump across statements.
- Strickly speaking, these are not allowed in structured programming.
- However, break and continue are especially useful.
- break and continue statements are used with while, for, and switch statements.
- break provides an exit condition other than that of the statement's test expression.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostrear brary

ontinuing our tou

ata Types

Expressions

Simple Statements

Scope

ooping Statements

do statement

o statement

Using a constant or singly variable as a test condition while and the null

tatement or Statement

Comparing while and

break, continue, goto

Functions

unctions

Pointore

. 0.....

., -

Example of use of continue

Kernighan and Ritchie provide this example of using continue:

See also my example program cat.cpp

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea ibrarv

ontinuing our tou

ata Types

Expressions

mple Statements

Scope

while Statement

while Statement

do statement

oid Confusing == with ing a constant or sing riable as a test condit

while and the null statement

for Statement

Comparing while and if and switch Statem

break, continue, goto

Functions

Pointers

Arrays and Pointers

goto: use it seldom

- goto and label statements allow the program to jump to any statement.
- Using goto can cause the program to become very hard to understand
- Use it only when you really have to
 - An example is to break out of a nested loop from the inner loop

```
 \begin{array}{ll} \text{for ( } i = small; \ i < big; \ ++i \ ) \\ & \text{for ( } j = small2; \ j < bigger; \ ++j \ ) \\ & \text{if ( } i == j \ ) \\ & \text{goto } \textit{equal}; \end{array}
```

equal:

equal here is a label

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream orary

ontinuing our tot

Expressions

Simple Statements

compound Statement Scope

oping Statements

hile Statement

do statement

oid Confusing == wi

while and the null statement

for Statement

Comparing while and for and switch Statements

break, continue, goto

Functions

Arrave

Pointers

Arrays and Pointe

Aultidimensional

Functions — 1

- A big program may be too complex to hold in your head.
- I need a way to break a big problem into many small, easy-to-understand problems.
- One way to break a problem into small problems is to divide a problem into small parts that can be written as functions.
- ▶ A *function* is a number of statements that:
 - Perform one easily-understood job
 - are given a single name.
- ➤ A function is a little bit like a simple IC, with input pins and output pins.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea brary

ontinuing our tou

ata Types

.

atements

Functions

Calling Functions
Using return Value for

unction Parameter

rays

ointers

rays and Pointers

ultidimensional

ructuros

oodina on

Guidalinas

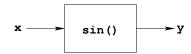
Guidelines

ADVABLATIVE FOOR

Some Things to Re

Functions — 2

A function may have inputs and outputs:



▶ The function call:

$$y = sin(x);$$

can be represented by the block diagram above.

- ► The inputs go in the parentheses: ()
- ▶ The output of the function can be assigned, as above.

C Programming for Engineers

Nick Urbanik

Who am I? Who an You?

Overview

C++ and iostrea: library

ontinuing our tou

ata Types

P ----

Statements

Functions

efining Functions alling Functions

Ising **return** Value from unctions

-unction ratameter

rays

ointers

ravs and Poin

lultidimensional

ays and array nters

Structures

Reading and Writi Files

Guidelines

Some Things to

Functions — 3

- ➤ To write and use functions in your program, there are two things to consider:
- When you define the function, you write the statements that the function will perform when it is called.
- When you want to use the function, we say we call the function.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overviev

C++ and iostrealibrary

onlinuing our to

ata Types

_xpi essions

tatements

Functions

Defining Functions
Calling Functions

Jsing return Value fro functions

Function Parameter

rays

ointers

avs and Poin

rrays and Pointers

tidimensional lys and arrays nters

Structures

leading and Writi iles

Guideline

ADVABLATIVE FOOR

Some Things to Re

All Operators and their Precedence

Functions **Defining Functions**

Arrays and Loops

C Programming for Engineers

Nick Urbanik

Defining Functions

slide 159/256

ADVABLATIVE FOOR

Function definition

- A function definition consists of a header and a body.
- header contains the function name, its return type, and parameter types.
- ► The body of a function is a block, which usually contains variable declarations and statements.

Example:

```
void calc( void )
{
     int num;
     num = 5;
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea brary

ntinuing our tour

a Types

Expressions

atements

Defining Functions

Calling Functions
Using return Value from
Functions

Function Parameter

rrays

Pointers

rrays and Poin

Arrays and Fornier

Multidimensiona Arrays and array pointers

Structures

Reading and Writ Files

Guidelines

Some Things to

All Operators and their Precedence

Functions

Calling Functions

Arrays and Loops

C Programming for Engineers Nick Urbanik

Calling Functions

Function definition and call

Program nowincal.cpp

```
// Program to call a function
#include < iostream>
// function definition:
void calc( void )
        std::cout ≪ "Now in Calc\n":
int main( void )
        std::cout ≪ "Hello World\n":
        calc();
        std::cout ≪ "Now in Main\n";
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

onlinuing our to

ata Types

Expi dodiono

Statements

unction

Defining Functions

Calling Functions

Jsing **return** Value from unctions

Function Parameters

rays

Pointers

rravs and F

Iriays and Folliters Iultidimonsional

Structures

Structures

eading and v les

Sudalina

Some Things to F

Who aim I r Who are Tou?
Overview
Quick Tour
Standard Input, Standard
Redirecting Output and
C++ and iost ream librar
An overview of the ios
library
Continuing our tour
hello.cpp
Pages Eventury

Input characters without skipping whitespace
I/O of other data to standard outpand from standard input

Reading into strings Loops Reading till the end of file

Data Types

Characters
Octal, Hexadecimal Output with

Floating Point Types Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence All Operators and their Precedence

Relational Expressions Logical Expressions Assignment Expression

Assignment Expressions
Increment, Decrement Operator

Arithmetic if Expressions
Bitwise Operators

atements

imple Statements compound Statements

Scope
Looping Statements
while Statement

do statement Avoid Confusing == with = Using a constant or single va

s a test condition

thile and the null statement

Comparing while and for

if and switch Statements

Functions

Calling Functions

Using return Value from Functions
Function Parameters

rrays Defining Arrays Arrays and Loops

Strings

Pointers

Arrays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Arrays of pointers Memory Allocation Multidimensional Arrays Command Line Argument

argv

Passing Structures to Function

eading and Writing Files

Error Handling
Binary files

Reading a Line at a time:

ADVABLATIVE FOOR

I/O of other data to/from Text File

Style Guidelines Program Design

Some Things to Read

C Programming for Engineers

Nick Urbanik

/ho am I? Who are ou?

Overview

C++ and iostream

ontinuing our tour

ata Types

.procolorio

atements

unctions Defining Functions

Calling Functions
Using return Value from

Functions
Function Parameters

unction Paramete

rays

Pointers

Arrays and Pointer

Multidimensiona Arrays and array

Structures

Reading and Wri

Some Things to

```
▶ Program calc-2.cpp
```

```
#include < iostream>
float calc( void )
        return 8.0 * 5.35;
int main( void )
        float res1 = calc():
        float res2 = 7 + 5 * calc():
        if (calc() > 5)
                std::cout << "Larger\n";</pre>
        std::cout \ll "res1 = " \ll res1
             \ll ", res2 = " \ll res2 \ll '\n':
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

O++ and iostream ibrary

Continuing our tou

ita Types

-Aprobolotio

Statements

Defining I

Calling Functions
Using return Value from

Functions
Function Parameters

Function Parameters

rrays

Pointers

rravs and

arrays and Pointer

Multidimensional Arrays and arrays pointers

Structures

leading and Wri iles

Guidelines

Some Things to I

return statement and function return value

- ► The return statement consists of the keyword return, an expression, and a semicolon.
- Syntax: return (expression);
- ► The ⟨*expression*⟩ is called the *return* expression.
- return statement will:
 - end the processing of a function
 - make execution continue from where the function was called, and
 - specify the function's return value.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our tour

ta Types

Expressions

atements

Defining Functions

Using return Value from Functions

unction Parameters

rrays

Pointers

ravs and Poir

ultidimensional rays and arrays

Structures

eading and Writ

Guidelines

ADVABLATIVE FOOR

Some Things to Re

Function return value: example

Program calc-3.cpp

```
#include < iostream>
  Example of a function that takes
// parameters
float calc( int num, float calc_rate )
    return calc_rate * num:
int main( void )
        float rate = 2.0;
        float res = calc(5, rate);
        std::cout \ll "res = " \ll res \ll ' \n':
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overvier

C++ and iostream

Continuing our tou

ata Types

Expressions

Statements

101101110

unction

efining Functions

Using **return** Value from Functions

Function Parameters

Arrays

ointers

rravs and F

.

ointers

Structures

Reading and Writing Files

.

Some Things to I

Return inconsistencies

- Return value inconsistencies: occur when the return expression has a type different from the function's return type.
- ► The return expression is what comes between the keyword return and the semicolon.
- The return type is what is written before the name of the function in a function definition.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrear library

ontinuing our tour

ata Types

Expressions

atements

Defining Function

Using **return** Value from Functions

unction Parameters

rays

ointers

..... D.

rrays and Pointers

Itidimensional ays and arrays nters

Structures

leading and Writi iles

Guideline

ADVABLATIVE SOC

Some Things to Re

ontinuing our tou

Expressions

Statementa

. .

unctions

Using **return** Value from Functions

Function Parameters

rrays

ointers

rrays and Pointe

Multidian and in the land

rrays and arrays of ointers

tructures

Reading and Writin Files

auidelines

Some Things to F

```
#include < iostream>
short getnum( void )
       long num = 2147483647L;
       std::cout ≪ "Number is " ≪ num ≪ '\n';
       return num;
int main( void )
       long res = getnum(); // return value inconsistency
       std::cout ≪ "result is " ≪ res ≪ '\n';
```

Output for program retbad-1.cpp:

Number is 2147483647 result is -1

Program retbad-2.c

```
#include < stdio.h>
calc( void ) /* return type missing */
                 /* gives return type inconsistency */
                 /* legal in C */
                 /* Illegal in C++, won't compile */
    float cost = 8.0 * 5.35:
    return cost;
int main( void )
    float res1;
    res1 = calc():
    printf( "%f\n", res1 );
```

► Output from program retbad-2.c:

42.000000

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ntinuing our tou

a Types

atements

Defining Functions

Using return Value from Functions

rravs

ninters

TOITILETS

rays and Pointe

ointers

structures

Reading and Writing Files

Guidelines

ADVABLATIVE FOOR

Some Things to Re

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

itinuing our to

hello.cpp Rasic Synt

Input characters without skipping

I/O of other data to standard output

and from standard input

Reading into strings

Loops
Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output with

Floating Po

Named Constants Enumerated Types: enum

Expressions

Arithmetic Expressions Arithmetic Operators and All Operators and their Precedence

Relational Expression

onical Expressions

Assignment Expressions
Increment Decrement Operator

Increment, Decrement Operator
Comma Operator

Arithmetic if Expressions
Bitwise Operators

tomonto

tements

ple Statements

Scope .ooping Statements

while Statement

do statement Avoid Confusing =:

Jsing a constant or single variates a test condition

while and the null statement

Comparing while and for

if and switch Statements

Functions

Calling Functions

Function Parameters

rave

Defining Arrays

Arrays and Loops

String

Pointers

Pointers as Function Parameter

ays and Pointer

Strong relationship between array and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation Multidimensional Arrays

Command Line Argum

Structures

Passing Structures to Functions

eading and Writing Files

fstream: file input and outp
Error Handling

Binary files Character I/O

Reading a Line at a time:

ADVABLATIVE FOOR

I/O of other data to/from Text Fil

Program Design

Some Things to Rea

Engineers Nick Urbanik

C Programming for

Vho am I? Who are

Overview

C++ and iostream

ontinuing our tour

ata Types

.procolorio

atements

unctions

efining Functions calling Functions

Functions

Function Parameters

rrays

Pointers

Arrays and Fornier

Multidimensiona Arrays and arra

Structures

Reading and Wri

Some Things t

slide 170/256

Function parameters

- Functions may have zero or more parameters.
- Parameters are usually the inputs to the function.
- Parameters appear in the parentheses after the name of the function, both in the function definition and in the function call.
- ▶ The *type* and *number* of parameters must match in:
 - function definition, and
 - function call

C Programming for Engineers

Nick Urbanik

Function Parameters

slide 171/256

ADVABLATIVE SOC

Function parameters — 2

In this example the parameters match

```
▶ Program funcmult.cpp
```

```
#include < iostream>
float mult( int a, float b)
    return a * b;
int main( void )
    int x = 3:
    float y = mult(x, 4.5); // function call
    std::cout \ll "y = " \ll y \ll ' n';
```

C Programming for Engineers

Nick Urbanik

Nho am I? Who are You?

Overview

C++ and iostream library

Continuing our to

ata Types

Expi dodiono

tatements

unctions

Defining F

alling Functions

Ising return Value fro

Function Parameters

runction ratameters

Tilays

Pointers

Arrays and F

fultidimensional

Structures

Reading and Writing Files

Guidelines

Some Things to

Function parameters — 3

▶ In the function definition of *mult*():

```
float mult( int a, float b )
{
    return a * b;
}
```

- ▶ the first parameter is called a.
- ▶ the second parameter is called *b*.
- In the function call,

```
float y = mult(x, 4.5); // function call
```

- ▶ the value of *x* is copied to *a*, the value 4.5 is copied into *b*.
- The type of the parameter in the function call matches the type of the parameter in the same position in the function definition.
- This is like pins on an IC plugging into the holes in the IC socket.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostrear library

munung our te

ta Types

Expressions

atements

unctions Defining Function

Calling Functions
Using return Value 1

Functions

Function Parameters

Function Parameters

rrays

Pointers

Arrays and Pointer

Multidimensional

Structures

eading and Wi

Guidelines

Some Things to

Arrays

Engineers Nick Urbanik

C Programming for

Who am I? Who are You?

Overview

O++ and iostream ibrary

.

statements

Arravs

Arrays

Arrays and Lo

Strings

omlers

Arrays and Pointe

ultidimensional rays and arrays of pinters

ructures

Reading and Writing

Guidelines

me Things to Read

An array is a collection of objects, all of the same data type.

- Any one data type can be used in an array.
 - an array of integers
 - an arrary of characters
 - an array of structures
 - an array of pointers
- The declaration of an array reserves memory, which is then managed by pointers (to be discussed in next section).
- Array objects themselves are actually referenced through pointer indirection.

All Operators and their Precedence

Arravs Defining Arrays

Arrays and Loops

C Programming for Engineers Nick Urbanik

Defining Arrays



Array declaration

- C Programming for Engineers Nick Urbanik

Defining Arrays

An array declaration consists of 4 parts

- the data type
- array name,
- square brackets around the...
- ... number of objects in the array

Arrays — 2

► The declaration below declares an array of 5 integers. The array name is *mynums*

```
int mynums[ 5 ];
```

Many different kinds of arrays may be declared, each having its own data type and number objects.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostre rary

ontinuing our tou

ata Types

unctions

rays

Defining Arrays

Strings

pinters

Arrays and Pointers

fultidimensional rrays and arrays of ointers

ructures

eading and Writing

uidelines

Array initialisation

C Programming for Engineers

Nick Urbanik

You?

Overview

C++ and iostream
brary

Continuing our tour

ata Types

....

unctions

rays

Defining Arrays Arrays and Loop

inters

rrays and Pointe

Autidimonsional

Itidimensional ays and arrays of nters

tructures

Reading and Writing

auidelines

me Things to Read

When a variable is defined it can be initialised with a value. In the declaration:

char mychar = 'E';

- ▶ the variable *mychar* is *initialised* with the character $' \, \mathbb{E}'$.
- ► The elements of an array can also be initialised in an array declaration. The initalisation values are listed within curly braces and separated by commas:

int $mynums[5] = {3, 4, 5, 6, 7,};$

++ and iosi orary

Continuing our tour

ata typoo

Function

Defining Arrays

efining Arrays

Strings

ointers

rrays and Pointe

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

Guidelines

me Things to Read

When using standard C, the initialization part of the array declaration can be used be left out of the array declarations. The number is, instead, determined by the number of values in the initialisation block.

```
► Program array-1.cpp
```

```
#include <iostream>
int main( void )
{
      char letters[] = { 'A', 'B', 'C', };
      int totals[] = { 23, 8, 11, 31, };
      std::cout << letters[ 1 ] << ' ';
      std::cout << totals[ 3 ] << '\n';
}</pre>
```

Array references and array notation

- Once an array has been declared, its objects can be referenced and used in expressions.
- ➤ The array name, together with the position of an object in the array is used to reference an object.
- The objects in an array are arranged in sequence, starting from zero. The number of an object's place in that sequence is referred to as either the object's index or subscript.
- In the example above, we can see

```
mynums[ 0 ] = 3; // first object
mynums[ 1 ] = 4; // second object
mynums[ 2 ] = 5; // third object
mynums[ 3 ] = 6; // fourth object
mynums[ 4 ] = 7; // fifth object
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our to

ila Types

.

unctions

rays

Defining Arrays

rrays and Loop:

ointers

Ulliters

Arrays and Point

rrays and arrays cointers

tructures

Reading and Writing Files

Guidelines



All Operators and their Precedence

Arravs

Arrays and Loops

C Programming for Engineers

Nick Urbanik

Arrays and Loops



Data Typos

Expressions

rave

Arrays and Loops

ointers

Arrays and Pointe

Multidimensional Arrays and arrays o pointers

tructures

Reading and Writing Files

Guidelines

```
Operations cannot be performed on an array as a whole.
```

- To assign a set of values to an array, you need to assign a value to each element individually.
- ➤ An array is only a collection of objects. It is not an object itself. We use loops to process all these objects.
- ▶ Program arrayprt.cpp

```
#include <iostream>
int main( void )
{
    int nums[] = { 23, 8, 11, 31 };
    for ( int i = 0; i < 4; ++i ) {
        std::cout ≪ nums[ i ] ≪ ' ';
    }
    std::cout ≪ '\n';
}</pre>
```

Array Management and Loops — 2

- ▶ A common rule of thumb is that the test for the end of an array is the less than operator, <, tested against the number of objects declared in the array.
- Example: this for loop prints each number in the array:

```
int nums[\ 4\ ] = \{\ 42,\ 1000,\ 7,\ 103\ \}; for ( int i=0;\ i<4;\ ++i ) std::cout \ll \text{"this num is "} \ll nums[\ i\ ] \ll ' \ '';
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

continuing our tour

ata Types

_xprocororio

tatements

rays

Defining Arrays

Arrays and Loops

Strings

inters

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

ructures

Reading and Writing Files

Guidelines

Using constants for array size

▶ In the listing below, the same symbolic constant, max, is used in both the array declaration and the test for the last array object in the for statement.

```
#include <iostream>
const int max = 4;
int main( void )
{
   int mynums[ max ] = { 23, 8, 11, 31 };
   for ( int i = 0; i < max; ++i )
        std::cout << mynums[ i ] << ' ';
        std::cout << '\n';
}</pre>
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream library

ontinuing our tour

ata Types

Expressions

Functions

Arravs

Defining Arrays

Arrays and Loops

trrays and Loops strings

ointers

Arrays and Pointer

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing Files

Guidelines

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostre

An overview of the iostream library

ontinuing our to

hello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard output

and from standard input

Reading into strings

Loops Reading till the end of fil

Infinite L

Integer Types

Octal, Hexadecimal Output with

ostream
Floating Point Typ

Named Constants
Enumerated Types: enum

Expressions

Arithmetic Expressions Arithmetic Operators and Precedence

All Operators and their Precedence

elational Expressions

Logical Expressions

Assignment Expressions
Increment, Decrement Operato

Comma Operator

Bitwise Operators

Statements

nple Statements

Scope

while Statement

do statement

woid Confusing == with = Ising a constant or single varial as a test condition

while and the null statement

Comparing while and for

f and switch Statements

Function

Calling Functions

Using return Value from Function Parameters

Arrays

Defining Arrays

Arrays and Loops Strings

Pointers

rs as Function Parameter

ays and Pointer

Strong relationship between array and pointers

Aultidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Arguments

Structures

Passing Structures to Functions

Reading and Writing Files

Error Handling Binary files

Reading a Line at a time:

I/O of other data to/from Text File

Guidelines

Program Design

Some Things to Rea

C Programming for Engineers Nick Urbanik

Overview

C++ and iostream

Continuing our tour

ata Types

tatoments

rrays

efining Array

Strings

nters

rrays and Pointe

Multidimensional Arrays and arrays pointers

Structures

Reading and Writing

Guidelines



ontinuing our tou

ata Types

Lxpressions

unctions

rrays Defining Array:

Defining Array Arrays and Lo

Strings

ointers

Arrays and Pointer

Multidimensional Arrays and arrays o pointers

tructures

Reading and Writing

Auidelines

- ► A *string* is an array of characters.
- Here are examples of definitions of strings:

```
char name[ 20 ];
char string[] = "This is a string";
char str[ 10 ] = "string";
char letters[ 10 ] = {
   's', 't', 'r', 'i', 'n', 'g', '\0'
};
```

- Note that the string definitions str and letters are equivalent.
- Note that a string is automatically ended with a special character called the *null* character, ' \0'

Arrays of characters: strings — 2

▶ Because the string has the extra '\0' character at the end, the array of characters must be long enough to hold it.

Example:

```
char string2[] = "string";
```

... has seven characters, so this would be wrong:

```
char string3[ 6 ] = "string"; // too short!
```

but these are okay:

```
char string4[ 7 ] = "string";
char string5[ 100 ] = "string";
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostrea orary

ontinuing our tou

ata Types

крі осологіо

atements

rrovo

Defining Arrays Arrays and Loops

Strings

ointers

Arrays and Pointer

Multidimensional arrays and arrays of cointers

tructures

Reading and Writing

Guidelines



Careful: strings cf. characters

- ▶ A common mistake is to confuse a string with a character.
- Example:

```
char c;
// Wrong! A character has single quotes: 'A'
c = "A";
```

- ► The string "A" is actually *two* characters: 'A' then '\0'.
- One final note: do not assign strings!

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream orary

nunuing our tour

ata Types

...

Defining Arrays

Strings

ointers

Arrays and Pointers

Multidimensional Arrays and arrays o pointers

tructures

Reading and Writing

Auidelines

Working with strings

► The standard C++ strings library is the best choice for simplicity, but the Borland 3.1 compiler does not seem to support it

- The standard library that comes with (nearly) every C compiler provides lots of functions for working with strings.
- ► To use them, put:

#include <string.h>

- at the top of your program.
- Here are some:
- strlen() give the length of a string
- strcpy() copy one string to another string
- strcmp() compare two strings
- strcat() join one string onto the end of another

C Programming for Engineers

Nick Urbanik

Vho am I? Who are fou?

Overview

++ and ios orary

Continuing our

ila Types

nctions

rays efining Arrays rrays and Loops

Strings

ointers

Arrays and Pointer

Multidimensional Arrays and arrays o pointers

Structures

Reading and Writing

Guidelines

Working with strings 2

Example using strlen():

```
int len;
len = strlen( "a string" ); // len = 8
```

Example using strcpy():

```
char str1[ 100 ], str2[] = "a string";
strcpy( str1, str2 );
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C<mark>++ and</mark> iostream l<mark>ibrary</mark>

ontinuing our toui

ta Types

expressions

_ ..

unctions

rays

lefining Arrays rrays and Loops

Strings

ointers

Arrays and Pointer

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing Files

Guidelines

Pointers

- Any object defined in a program can be referenced through its address. A pointer is a variable that has as its value the address of an object.
- A pointer is used as a referencing mechanism.
- A pointer provides a way to reference an object using that object's address.
- ▶ There are 3 elements involved in this referencing process:
 - a pointer variable
 - an address
 - and another variable

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

O<mark>++ and</mark> iostream ibrary

Continuing our tour

ata Types

.

rrave

irrays

Pointers

Pointers as Function Parameters

rrays and Pointers

ultidimensional

Structures

Reading and Writing Files

Guideline



Pointers

A pointer variable holds the address of another variable of a particular type

▶ **Program** pointer-1.cpp

```
#include < iostream>
int main( void )
   int num = 12;
   int *nptr;
   nptr = #
   std::cout ≪ "num holds " ≪ num
        \ll " and nptr points to " \ll *nptr \ll '\n';
   std::cout \ll "The address held in nptr is " \ll nptr
```

▶ Output of program pointer-1.cpp:

num holds 12 and nptr points to 12 The address held in nptr is 0xbfafade8

All Operators and their Precedence

Arrays and Loops

Pointers

Pointers as Function Parameters

Engineers Nick Urbanik

C Programming for

Pointers as Function Parameters



```
void swap( int x, int y )
{
     int temp = x;
     x = y;
     y = temp;
}
```

and call it like this: swap(a, b);, the values of a and b are copied by value, so the final values are not changed.

► The right way is to pass the *address* of *a* and *b* like this: *swap*(&*a*, &*b*); and define the function like this:

```
void swap( int *x, int *y )
{
    int temp = *x;
    *x = *y;
    *y = temp;
}
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

verview

C++ and iostrea ibrary

ontinuing our tour

ata Types

-xpi 03310113

Eunotione

Arrays

ointers

Pointers as Function Parameters

Arrays and Pointers

ultidimensional rrays and arrays of

Structures

Reading and Writi

Guidelines

Some Things to Read

► This is how to change the value of a parameter.

All Operators and their Precedence

Arrays and Loops

Arrays and Pointers Strong relationship between arrays

and pointers

C Programming for Engineers

Nick Urbanik

Strong relationship between

arrays and pointers



- Pointers and arrays have a strong relationship
- Any operation using array subscripting can be done using pointers
- ▶ If we define an array of integers and a pointer to an integer like this:

```
int a[ 10 ];
int *pa:
```

and if we make pa point to the start of the array a[] like this:

```
pa = &a[0];
```

then the value of *pa is the same as is stored in a[0].

- ▶ The pointer pa + 1 points to the value of a[1], so this statement is true: *(pa + 1) == a[1]
- If we add '1' to a pointer, we point to the address just after the value stored at that pointer.
 - If sizeof(int) is 4, and if addresses each hold one character, then the address pa + 1 is 4 address locations higher than the address pa.
 - ► The *type* of the pointer determines what address you get when you increment a pointer.
- \triangleright similarly *(pa + i) == a[i] is true.

C Programming for Engineers

Nick Urbanik

Strong relationship between arrays and pointers

Strong relationship between arrays and pointers

The name of an array is the same as the location of the first element, so these two statements are equivalent:

```
pa = &a[0];
pa = a:
```

These statements are also true:

```
a[i] == *(a + i);
&a[ i ] == a + i;
```

- An expression made of an array and index has an equivalent expression made with a pointer and offset.
- Important: we can do

```
++pa; // okay; now pa points to a[1]
```

... but not:

... because an array name is a constant, but an ordinary pointer is not.

Passing arrays to functions

- If a parameter is an array name, inside the function it is a pointer to the first element of the array
- If you find the size of an array with sizeof, you are given the number of elements × the size of one element
- Inside a function, the size of an array parameter is the size of a pointer.
- ► See program array-parameter.cpp.
- Note that the sizeof operator gives the number of characters in either a type or an expression:
- sizeof(T) gives the number of characters in the type T
- sizeof expression gives the number of characters in the expression expression.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

verview

++ and i

Continuing our tou

ata Types

.,.....

inctions

rrays

Strong relationship between arrays and pointers

Multidimensional Arrays and arrays pointers

Structures

Reading and Writing Files

auidelines



Program array-parameter.cpp

```
#include < iostream>
int ar[10] = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\};
void check_array_parameter( int a[] )
       std::cout \ll "sizeof(a) = " \ll sizeof(a) \ll ' n';
       std::cout \ll "sizeof(ar) = " \ll sizeof(ar) \ll ' n';
int main( void )
       check_array_parameter( ar );
       int nelements = sizeof( ar ) / sizeof ar[ 0 ];
       std::cout \ll "sizeof(ar) = " \ll sizeof(ar) \ll ' n'
            \ll "number of elements in ar[] is " \ll nelements
```

▶ Output of array-parameter.cpp

```
sizeof( a ) = 4
sizeof( ar ) = 40
sizeof( ar ) = 40
number of elements in ar[] is 10
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ontinuing our tour

ita Types

Expressions

Statements

rrovo

. .

annua and Daintana

Arrays and Pointers
Strong relationship between arrays and pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing Files

Guidelines

Continuing our tour

Data Types

_

- unction

Pointore

Arroya and Paint

Arrays and Pointers
Strong relationship between

arrays and pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing Files

Guidelines

Some Things to Read

► To work properly, you need to pass the length of an array as a separate parameter together with the array.

Example:

```
double mean( int nums[], int len )
{
    int sum = 0;
    for ( int i = 0; i < len; ++i ) {
        sum += nums[ i ];
    }
    if ( len == 0 ) return 0;
    return ( double ) sum / len;
}</pre>
```

- Note that it makes no difference to write the parameter as int nums[100], since that length information will not be passed to the function as part of the int nums[100] parameter
- ▶ If the length of the array passed to *mean*() is 100, then that number must be passed as a *separate parameter*.

All Operators and their Precedence

Arrays and Loops

Multidimensional Arrays and arrays of pointers

Arrays of pointers

C Programming for Engineers

Nick Urbanik

Arrays of pointers

arac, arav

Arrays of pointers

- Arrays of pointers are very commonly used in C, because this gives much greater flexibility than alternatives (see slide §216)
- Can easily sort an array of pointers; copy only the address, not the data
- Can define an array of pointers to lines like this:

```
const int maxlines 10000;
char *line[ maxlines ];
```

We must make sure that we allocate memory using the new operator for this as we read lines.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iosibrary

ontinuing our toui

ata Types

......

Functions

Arrays

Arrays and Pointers

lultidimensional rrays and arrays of pinters

Arrays of pointers Memory Allocation

Multidimensional Arrays
Command Line Arguments
argc, argv

Structures

eading and Writ iles

Guidelines

ADVABLATIVE FOOR

Some Things to F

All Operators and their Precedence

Arrays and Loops

Multidimensional Arrays and arrays of

pointers

Memory Allocation

C Programming for Engineers

Nick Urbanik

Memory Allocation

arac, arav

Allocating memory with **new**

- If we don't know how much data we will read, we need to allocate memory as we need it
 - The new operator allocates memory as it is needed
 - deallocate (free) memory with the delete operator
- If allocating a scalar value, use syntax like this:

```
\langle pointer \rangle = \mathbf{new} \langle type \rangle;
and free the memory like this:
delete \(\(\rhointer\);
```

if allocating an array or string, use syntax like this:

```
\langle pointer \rangle = new \langle type \rangle [ \langle length \rangle ];
and free the memory like this:
```

```
delete [] \(\langle pointer \rangle;
```

- ▶ the result from **new** is a null pointer if the memory cannot be allocated.
 - Always check the return value of new

C Programming for Engineers

Nick Urbanik

Memory Allocation

Example use of new: Program new-1.cpp

- ▶ Program new-1.cpp does the following:
 - dynamically allocate an array of ten integers
 - terminate if allocation doesn't succeed.
 - put a value into each element of the array
 - print each value

```
free up the memory
```

```
#include < iostream>
#include < stdlib.h>
int main( void )
       const int maxn = 10:
       int *a = new int[ maxn ];
       if (a == NULL)
               std::cerr ≪ "Out of memory!\n";
               exit( 1 ):
       for ( int i = 0; i < maxn; ++i )
               a[i] = i + 1;
       for ( int i = 0; i < maxn; ++i )
               std::cout \ll "a[" \ll i \ll "] = "
                    \ll a[i] \ll ' n':
       delete [] a;
```

C Programming for Engineers

Nick Urbanik

Memory Allocation

arac, arav

Program new.cpp — 1

```
#include < iostream>
#include <string.h>
// Read all of input into memory.
                                     Normally we would process one line
// at a time, as we read it.
// Read each line into a string
// allocate memory for the string and copy the string into that memory
// add the newly allocated string to an array of pointers
int read_lines( char *lines[]. int maxnlines )
        const int maxlinelen = 8000:
        char line[ maxlinelen ]:
        int nlines = 0:
        while ( std::cin.getline( line, maxlinelen ) ) {
                 int len = std::cin.acount(): // includes space for '\0'
                 char *p;
                 if ( nlines >= maxnlines
                      | | (p = new char[len]) == NULL) 
                         return -1;
                 } else {
                          strcpv( p, line );
                         lines[ nlines++ 1 = p;
        return nlines:
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ontinuing our tour

Data Types

P ----

Statements

urays

..... and Date

Multidimensional

Arrays of pointers

Memory Allocation

Command Line A

Structures

Reading and Writing Files

Guidelines

ADVABLATIVE FOOR

Program new.cpp — 2

```
void write_lines( char *lines[], int nlines )
        while ( nlines-->0 )
                std::cout << *lines++ << '\n':
void free_lines( char *lines[], int nlines )
        while ( nlines-->0 )
                delete [] *lines++;
int main( void )
        const int maxlines = 10000:
        char *line[ maxlines ]:
        int nlines = read_lines( line, maxlines );
        if (nlines >= 0) {
                write_lines( line, nlines );
                free_lines( line, nlines ):
        } else -
                std::cerr << "Input is too big to read\n";</pre>
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Worview

C++ and iostream

Continuing our tou

ta Types

Expressions

Statements

IIays

0.....

Arrays and Pointers

Multidimensional Arrays and arrays of

Arrays of pointers

Memory Allocation

Command Line A argc, argv

Structures

Reading and Writing Files

Guidelines

Some Things to F

All Operators and their Precedence

Arrays and Loops

Multidimensional Arrays and arrays of pointers

Multidimensional Arrays

C Programming for Engineers

Nick Urbanik

Multidimensional Arrays

Multidimensional Arrays

- Not used as much as arrays of pointers
- Usually we allocate memory with new for each string

defines a rectangular matrix.

We can access the entry with the value 30 with

```
int entry = matrix[1][2];
```

▶ Note that this is *wrong*, and just uses the comma operator:

```
int entry = matrix[ 1, 2 ]; // WRONG!!!
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream orary

ontinuing our tour

ta Types

Arrays

pinters

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

rrays of pointers

Memory Allocati

Multidimensional Arrays Command Line Argume

Structures

Reading and

Guidelines

Some Things to

%

All Operators and their Precedence

Arrays and Loops

Multidimensional Arrays and arrays of pointers

Command Line Arguments: argc, argv

ADVABLATIVE FOOR

C Programming for Engineers

Nick Urbanik

Command Line Arguments: arac, aray

argc and argv

▶ The *main()* function takes two optional parameters that are always called argc and argv:

```
int main( int argc, char *argv[])
```

- parameter argc is the number of arguments on the command line *including the program name*
- parameter argy is a pointer to an array of command line arguments
- ▶ if the program echo is called like this:

```
echo this is a test
```

▶ then argc is 5, argv[0] is "echo", argv[1] is "this", argv[2]is "is", argv[3]is "a", argv[4]is "test" and finally argv[5] is the null pointer.

C Programming for Engineers

Nick Urbanik

Command Line Arguments: arac, arav

Program echo.cpp

▶ The program echo.cpp:

▶ prints its parameters, like the echo command in the Windows CMD. EXE shell, or like the echo command built into the bash shell that is popular with Linux.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

continuing our tour

a Types

_xp1000i0i10

Functions

rrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Memory Allocation
Multidimensional A

Command Line Arguments: argc. argv

Structures

Reading and Writ

Guidelines

Some Things to

- A tag can be used to label a structure type declaration. In the structure type declaration, the keyword struct may be followed with a tag placed before the opening brace of the declaration list,
- Example:

```
struct employee {
    int id;
    float salary;
};
```

Who am I? Who are You?

Overview

++ and iostream orary

ontinuing our tou

ata Types

Expressions

unctions

rrays

inters

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Passing Structures to Functions

typedef

Reading and Writing Files

Guideline

Defining struct variables

We can define a variable person of the type struct employee like this:

```
struct employee {
    int id;
    float salary;
};
// somewhere else:
struct employee person;
```

We can now refer to the two values in the variable person as person.id and person.salary

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

verview

++ and iostream orary

ontinuing our tou

a Types

Expressions

unctions

rave

--, -

.

rrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Passing Structures to Functions

/реает

Reading and Writing Files

Guidelines

Initialising struct variables

- We can initialise a structure when it is defined by putting a list of values in braces, as we do for arrays
- The first item in that list initialises the first element of the structure,
- the second item initialises the second element of the structure,
- We could initialise our struct employee like this:

```
struct employee person = { 8, 80000 };
```

After this, person will contain the same values as in the next slide.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

-+ and iostrearary

ontinuing our tour

ila Types

.

unctions

rays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Passing Structures to Functions

/pedef

Reading and Writing Files

Guidelines



Structures: Program struct.cpp

```
#include < iostream>
int main( void )
       struct employee {
               int id;
               float salary;
       struct employee person;
       person.id = 8:
       person.salary = 80000;
       std::cout << "ID = " << person.id << '\n';
       std::cout ≪ "Salary = $" ≪ person.salary ≪ '\n';
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

verview

C++ and iostrea ibrary

ontinuing our tour

a Types

Expressions

unctions

rravs

ninters

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Passing Structures Functions

typedef

Reading and Writing Files

Guidelines

All Operators and their Precedence

Arrays and Loops

Structures

Passing Structures to Functions

Engineers Nick Urbanik

C Programming for

Passing Structures to Functions

typedef



```
struct employee {
    int id;
    float salary;
};
struct employee person;
struct employee *p = &person;
```

we could access the members with the arrow operator '->' like this:

```
\begin{array}{ll} p->id = 8; \\ p->salary = 80000; \\ std::cout \ll "ID = " \ll p->id \ll ' \n'; \\ std::cout \ll "Salary = $" \ll p->salary \ll ' \n'; \\ \end{array}
```

- ► There is nothing magic about the "->" operator; it is just a shorthand used, because we often access members of structures through pointers
- Note that "p—>id" is equivalent to "(*p).id", and "p—>salary" is equivalent to "(*p).salary".

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostrea: library

ntinuing our toui

ata Types

Statamenta

Functions

rrays

pinters

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Passing Structures to Functions

Reading and Writing

Guidelines

Passing Structures to Functions

- ▶ Unlike arrays, structures are passed to functions by value
- That means that your function only gets a copy of the structure.
- If you want to modify the original structure, you need to either:
 - return the modified structure, or
 - pass a pointer to the structure.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our tou

ta Types

Expressions

rrovo

--, -

IIICIO

Arrays and Pointers

fultidimensional rrays and arrays of ointers

Structure

Passing Structures to Functions

typedef

Reading and Writin Files

Guidelines



```
struct employee raise_salary( struct employee p, float raise )
     p.salary += raise;
     return p;
// in another function:
struct employee manager = { 50, 100000 };
manager = raise_salary( manager, 20000 );
Passing a pointer to structure:
void raise_salary( struct employee *p, float raise )
     p \rightarrow salary += raise;
// in another function:
struct employee manager = { 50, 100000 };
raise_salary( &manager, 20000 );
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

verview

-+ <mark>and</mark> iost:

ontinuing our tou

a Types

Kpi essions

Statements

. ayo

JIIILEIS

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Passing Structures to Functions

typedef

Reading and Wri Files

Guidelines

C++ and iostreas

Continuing our tou

ata Types

Expression

Statements

Functions

Arrays

ointers

rrays and Pointe

Multidimensional Arrays and arrays of

Structure

Passing Structures to Functions

typedef

Reading and Writin Files

Guidelines

Some Things to Read

```
#include < iostream>
struct complex {
        int re:
        int im:
};
complex cadd( complex z1, complex z2 )
        complex zt;
        zt.re = z1.re + z2.re;
        zt.im = z1.im + z2.im:
        return zt:
int main( void )
        complex za:
        za.re = 1;
        za.im = 2:
        complex zb = za;
        complex zc = cadd(za, zb);
        std::cout << "zc.re = " << zc.re</pre>
             \ll ", zc.im = " \ll zc.im \ll '\n';
```

Output of program complex.cpp:

zc.re = 2, zc.im = 4

◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ◆ ◆ ○ ◆

All Operators and their Precedence

Arrays and Loops

Structures

typedef

Engineers Nick Urbanik

C Programming for

typedef



typedef

A typedef is used to allow the programmer to give another name to a type.

▶ typedef (type) (NAME); defines (NAME) as a new name for the existing type (type)

► Example:

```
typedef float FF; // FF is now a type FF abc; // equivalent to float abc;
```

- Often typedef is used with struct variables to avoid needing to type the word "struct"
 - I do not encourage you to do this.
 - See

```
http://www.kroah.com/linux/talks/ols_2002_kernel_codingstyle_paper/codingstyle.ps, and search for "typedef is evil" at section 3.5.
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

verview

C++ and iostream brary

ontinuing our tour

ta Types

- xp1 00010110

unctions

Arrays

ointoro

Arrays and Pointers

Multidimensional arrays and arrays of cointers

Structures
Passing Structures

typedef

Reading and Writin

Reading and writing Files

Guideline

All Operators and their Precedence

Arrays and Loops

Reading and Writing Files fstream: file input and output

C Programming for Engineers Nick Urbanik

fstream: file input and tuatuo

Text Files — ifstream and ofstream

- We work with text files much the same way as with standard input and standard output
- ► Open a file for input ifstream
- ▶ Open a file for output ofstream
- when the fstream object goes out of scope, the file is automatically closed

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostrea library

ontinuing our tou

ata Types

LXPI C33I0II3

unctions

Arravs

irrays

nters

rrays and Pointers

Multidimensional Arrays and arrays of

ructures

Reading and Writing

fstream: file input and output

Error Handling

Character I/O

Reading a Line at a time

of other data at Files

Who am I? Who are You

Quick Tour

Standard Input, Standard Output Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ntinuing our to

hello.cpp Rasic Synta

Input characters without skipping

I/O of other data to standard outp

and from standard input

Reading into strings

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output with

ostream Floating Poi

Named Constants

Expressions

Arithmetic Expressions Arithmetic Operators and

Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions

Assignment Expressions
Increment, Decrement Operators

Comma Operator

Bitwise Operators

atements

ements

ple Statements
npound Statements

_ooping Statement

while Statement

Avoid Confusing == with =
Jsing a constant or single varial

as a test condition

while and the null statement

for Statement Comparing while and for

if and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

String

Pointers

Pointers as Function Paramete

ays and Pointer

and pointers

Multidimensional Arrays and arrays o

pointers

Memory Allocation

Multidimensional Arrays

Command Line Arguments

Structures

Passing Structures to Function: typedef

Reading and Writing Files

fstream: file input and outp

Binary files

Reading a Line at a time:
getline()

Guidelines

Style Guidelin

Modules

Some Things to Rea

C Programming for Engineers Nick Urbanik

ho am I? Who are

Overview

C++ and iostrea: library

ontinuing our tour

ta Types

prossions

.......

rrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays o

Structures

Reading and Writing Files

fstream: file input and

Error Handling

Binary files

Reading a Line at a time

Text Files

If cannot open a file, what next?

- If you fail to create an ifstream object (perhaps the file does not exist), the object evaluates as false
- It is always essential to test the result of things that can go wrong, and provide an error message or return an error code
- ▶ If things have gone wrong, it may be better to stop the program using the *exit*() command.
- ▶ Note: **#include** <*stdlib.h*> when you use *exit*().

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and ios orary

Continuing our tour

ata Types

Function

Arrays

ointore

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing

Files

output

Error Handling

Binary files

Reading a Line at a time

O of other data

Text Files — 7

```
#include < iostream>
#include <fstream>
// . . .
std::ifstream fin( "data.in" );
if (! fin ) {
        std::cerr ≪ "error: unable to open file "
             << "'data.in' for reading\n";</pre>
        exit( 1 );
std::ofstream fout( "data.out" );
if (! fout ) {
        std::cerr ≪ "error: unable to open file "
             << "'data.out' for writing\n";</pre>
        exit( 1 );
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostread library

ontinuing our tour

ata Types

Expressions

Actionicino

unotions

Arrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing

fstream: file input and

error Handling

Error Handling

Character I/O

getline()
I/O of other data to/from

► To append to a file, use the extra parameter std::ios::app when defining the ofstream object:

- ► There are a number of other parameters available.
- ▶ OR them together with the bitwise OR operator "|"

Vho am I? Who are

Overview

++ and iostream rary

ontinuing our tour

ata Types

.

inctions

rays

Olliters

Multidimensional

Multidimensional Arrays and arrays o pointers

Structures

Reading and Writing

fstream: file input and

Error Handling

Binary files Character I/O

Reading a Line at a time:

of other data t xt Files

Who am I? Who are You'

Quick Tour

Standard Input, Standard Output
Redirecting Output and Input

C++ and iostream library

An overview of the iostream library

ontinuing our to

nello.cpp Basic Synta

Input characters without skipping

I/O of other data to standard output

and from standard inp

Reading into strin

Reading till the end of file

Data Types

Integer Types

Octal, Hexadecimal Output with

ostream Floating Poi

Named Constants

Expressions

Arithmetic Expressions
Arithmetic Operators and

Precedence

All Operators and their Precedence

Relational Expressions

ogical Expressions

Assignment Expressions
Increment, Decrement Operator

Increment, Decrement Operato Comma Operator

Arithmetic if Expressions

Dasts

atements

ple Statements

Scope

while Statement

Avoid Confusing == with = Jsing a constant or single variab

while and the null statement

for Statement
Comparing while and for

if and switch Statements

Functions

Defining Functions
Calling Functions

Using return Value from Function

Arrays

Defining Arrays

Arrays and Loops

String

Pointers

Pointers as Function Paramete

ays and Pointer

and pointers

Iultidimensional Arrays and arrays o

pointers

Memory Allocation
Multidimensional Arrays

Command Line Arguments:

Structures

ssing Structures to Functions

Reading and Writing Files

fstream: file input and outp

Binary files

Reading a Line at a time:

Guidelines

Style Guidelines Program Design

Some Things to Rea

Engineers Nick Urbanik

C Programming for

ho am I? Who are

Overview

C++ and iostrea library

ontinuing our tour

na Types

atomonte

unctions

rays

Arrave and Pointors

Multidimensional

ounters

Structures

Reading and Writin Files

fstream: file input and output

Error Handling

Binary files

Character I/O
Reading a Line at a t

O of other data ext Files

C++ and iostrea ibrary

ontinuing our to

ata Types

-xpi 03310113

Function

ırrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of

Structures

Reading and Writing

fstream: file input; output

Error Handling

Binary files

Reading a Line at a time: getline()

► To open a file for binary input or output, use the extra parameter std::ios::binary when defining the ofstream or ifstream object:

Here we open a binary file for input:

Binary files — 2

► Here we open a binary file for output:

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream

ontinuing our tou

ta Types

Expressions

atements

Arrays and Pointers

Multidimensional

Structures

leading and Writing

Files

output Error Handling

Binary files

Reading a Line at a time:

Text Files

All Operators and their Precedence

Arrays and Loops

Reading and Writing Files

Character I/O

C Programming for Engineers Nick Urbanik

fstream: file input and

Character I/O

Character Input and Output

- ▶ We can treat a file as a stream of characters. The istream member function get() and the ostream member function put() read and write one character at a time.
- ▶ Program copy-file-to-output.cpp:

```
#include < iostream>
#include < fstream>
#include < stdlib.h>
int main( void )
        char ch:
        std::ifstream fin( "abc.txt" );
        if (! fin ) {
                 std::cerr << "Cannot open file abc.txt\n":</pre>
                 exit( 1 );
        // skips whitespace.
        // while ( fin \gg ch )
                 std::cout << ch;
        while ( fin.get( ch ) )
                 std::cout.put( ch );
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostre orary

ontinuing our tou

ta Types

_xp1000i0i10

unctions

rays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays o pointers

Structures

Reading and Writing Files

output Fror Handling

Binary files

Character I/O

Reading a Line at a time: getline() I/O of other data to/from

Character I/O with Text Files — 2

► Program copyfile.cpp

```
#include <iostream>
#include <fstream>
#include < stdlib.h>
int main( void )
        char ch;
        std::ifstream fin( "abc.txt" );
        std::ofstream fout( "mmm.txt" );
        if (! fin || ! fout ) {
            std::cerr ≪ "Problem opening files\n";
            exit( 1 );
        // The following skips white space:
        // while ( fin \gg ch )
        //
              fout \ll ch;
        while ( fin.get( ch ) )
                fout.put( ch );
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostrea library

ontinuing our to

ta Types

_xp1000i0i10

unctions

rrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing

fstream: file input a

Error Handling

Character I/O

Reading a Line at a ti

O of other data

All Operators and their Precedence

Arrays and Loops

Reading and Writing Files

Reading a Line at a time:

getline()

C Programming for Engineers

Nick Urbanik

fstream: file input and

Reading a Line at a time: getline()

Working with Lines in Text Files

- We often want to work with one line of a text file at a time
- the istream member function getline() reads from a file and places it in a string or character array, without the ending newline.
 - $std::cin.getline \left(\ \left\langle string \ or \ character \ array \right\rangle, \ \left\langle maximum \ length \right\rangle \ \right) \ ;$
- ▶ getline() returns false if there is any error.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream ibrary

munung our te

ata Types

expressions

olatements

rrovo

.

omers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

ructures

Reading and Writing

Files

output Error Handling

Binary files

Reading a Line at a time: getline()

I/O of other data to Text Files

```
C Programming for 
Engineers
```

Nick Urbanik

```
Who am I? Who are You?
```

Overview

C++ and iostream library

Continuing our i

ta Types

Expressions

_ ..

irravs

ointers

Multidimensional

lultidimensional rrays and arrays of ointers

Structures

Reading and Writing

Files
fstream: file input and

output Error Handling

inary files

Reading a Line at a time: getline()

```
#include < iostream>
#include < fstream>
#include < stdlib.h>
int main( void )
        const int maxline = 100;
        char line[ maxline ];
        std::ifstream fin( "abc.txt" );
        if (! fin ) {
                std::cerr ≪ "Cannot open file abc.txt\n";
                exit( 1 );
        while ( fin.getline( line, maxline ) )
                std::cout << line << '\n':
```

All Operators and their Precedence

Arrays and Loops

Reading and Writing Files

I/O of other data to/from Text Files

C Programming for Engineers

Nick Urbanik

fstream: file input and

I/O of other data to/from Text Files

Reading other data from Text Files — 1

We can write formatted text to text files just as we can to standard output with std::cout

```
▶ Program fileout.cpp
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream brary

ontinuing our tou

ta Types

Expressions

unctions

rrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays o pointers

Structure

Reading and Writing

fstream: file inpoutput

Error Handling

Error Handling Binary files

haracter I/O

I/O of other data to/from Text Files

Reading other data from Text Files — 2

- ► We can read formatted text from text files just as we can from standard input with *std*::*cin*
- ▶ Program filein.cpp:

```
#include < iostream>
#include <fstream>
#include < stdlib.h>
int main( void )
       int num;
        std::ifstream fin( "abc.txt" );
       if (! fin ) {
               std::cerr ≪ "Cannot open abc.txt\n";
               exit( 1 );
       fin \gg num;
        std::cout << "we got '" << num
            \ll "' from abc.txt\n":
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

C++ and iostream

ontinuing our tour

ata Types

......

unctions

rrays

Pointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing

fstream: file inpi output Error Hondling

Error Handling

Binary files Character I/O

I/O of other data to/from

All Operators and their Precedence

Arrays and Loops

Guidelines

Style Guidelines

C Programming for Engineers Nick Urbanik

Style Guidelines

Modules

Style Guidelines

Use spaces after commas, around operators,

Example:

```
printf( "%d", i );
not
printf("%d",i);
and
x = x + 3:
not
x = x + 3;
```

- I suggest you put your main() function last.
 - avoids the need to put "function prototypes" that need unnecessary extra maintainance
- Use modern books about C, not very old ones.
- Indent your program to make it easy to follow.
- Indent the body of loops and if statements.

All Operators and their Precedence

Arrays and Loops

Guidelines

Program Design

Engineers Nick Urbanik

C Programming for

Program Design

Modules

library

Data Taras

expressions

Arrays

inters

Arrays and Pointers

Multidimensional Arrays and arrays o pointers

ructures

Reading and Writing

iuidelines

Style Guidelines
Program Design

Aodules

Some Things to Re

"Real" programs need to be designed; they may be too complicated to hold in your head all at one time.

- Sitting at the keyboard and typing a program as you think it up may work for small programs, but bigger programs written this way will become very messy and expensive to maintain.
 - The result is rather like a rough sketch to try out ideas.
 - You may want to start again after the experience you get from this "sketch"
- ► The greatest cost for a program is usually in maintaining it.
- Flowcharts:
 - are okay for simple programs
 - are good for representing complicated, unstructured looping and branching
 - ▶ **But:** a flowchart can easily become more complicated than the program itself!

Pseudocode: a basic design tool

- Pseudocode is a more practical way to show how your program will work.
- Pseudocode is a mixture of: English and structured C programming statements, such as if, while, do.
- Pseudocode should be simpler and easier to understand than the final program source code
- Pseudocode should be written before you type in your new program!

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

++ and iostream brary

ontinuing our tour

ta Types

atomonto

rrave

Arrays and Pointers

lultidimensional rrays and arrays of pinters

tructures

Reading and Writing Tiles

Guidelines

Style Guidelines
Program Design

Modules

Some Things

Program design: top-down

- Top-down design looks at the big picture first, forgetting the details.
- Write pseudocode including only these important, big steps, leaving out small steps at first. This is like your main() function.
- Write more pseudocode for each of these big steps, giving more detail, but not the smallest details. These are written rather like function definitions.
- For each step, write out more pseudocode like more function definitions, until you have enough detail to begin writing your source code.
- Check your design carefully before you move on.

C Programming for Engineers

Nick Urbanik

Who am I? Who are

verview

C++ and io brary

Continuing our to

ata Types

unctions

Arrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing Files

uidelines

Style Guidelines
Program Design

Modules

Some Things to



All Operators and their Precedence

Arrays and Loops

Guidelines

Modules

Engineers Nick Urbanik

C Programming for

Modules

Making an application from many modules

C Programming for Engineers Nick Urbanik

Who am I? Who are

Overview

orary

. _

Expressions

Function

Arrays

Pointore

Arrays and Pointe

Multidimensional

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

uidelines

tyle Guidelines rogram Design

Modules

Some Things to Read

 Most useful C or C++ programs are written in separate modules

- Each module corresponds to:
 - one .c, .C, .cc, .cpp, or .cxx file (lets call this the source file), and
 - ▶ one .h, .H, .hh, or .hpp file (we call this the header file).
- Each header file lists the publicly exported names: type definitions, global variables, and function prototypes
 - Avoid defining variables or functions in header files
- It makes this set of names as small as possible to reduce the interaction between modules
- All non-public functions defined in the source files are defined with the keyword **static** so that they cannot be linked to from other modules.

```
C Programming for
   Engineers
```

Nick Urbanik

Modules

```
#include < iostream>
#include "calc.h"
#include "main.h"
int glob;
int main( void )
       alob = 10;
       int sum = calc(15);
       std::cout \ll "sum = " \ll sum \ll ' \n';
main.h:
#ifndef MAIN H
#define MAIN_H
```

extern int glob;

Modules — silly example (continued) calc.cpp:

```
#include "main.h"
#include "calc.h"
int calc( int n )
       return glob + n;
calc.h:
#ifndef CALC_H
#define CALC H
extern int calc(int n);
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ntinuing our tour

ata Types

expressions

Statements

arays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

Structures

Reading and Writing Files

Guidelines

Style Guidelines

Modules

#endif

- the static keyword can (and should) be used with functions that are to be used only within one file
- ▶ The static keyword can be used inside functions and blocks
 - It is initialised once, and its value remains even between function calls. See Program static.cpp:

▶ output of static.cpp:

```
called 1 times called 2 times called 3 times called 4 times called 5 times
```

C Programming for Engineers

Nick Urbanik

Who am I? Who are

Overview

++ and iostream brary

ntinuing our toui

ta Types

xpressions

Lunctions

rrays

intore

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing Files

Guidelines

Style Guidelines Program Design

Modules

Resources I

Brian W. Kernighan and Dennis M. Ritchie. The C Programming Language.
Proprise Use 1088

Prentice Hall 1988.

Stanley B. Lippman and Josée Lajoie and Barbara E. Moo. C++ Primer, Fourth Edition. Addison-Wesley 2005.

Bjarne Stroustrup. The C++ Programming Language (Special 3rd Edition). Addison-Wesley, 2004, ISBN 0201889544.

Tom Adamson and James L. Antonakos and Kenneth C. Mansfield Jr. Structured C for Engineering and Technology, Third Edition.

Prentice Hall, 1998.

Steve Oualline. Practical C Programming. O'Reilly 1993. C Programming for Engineers

Nick Urbanik

Vho am I? Who are

Overview

D++ and lostream brary

ontinuing our tour

ta Types

_xpressions

Dialements

.....

..., -

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

ructures

Reading and Writing Files

auidelines

Resources II

Paul Davies

The Indispensable Guide to C With Engineering Applications.

Addison-Wesley 1995.

H. M. Deitel and P. J. Deitel. C How to Program, Second Edition. Prentice Hall 1994.

Vincent Kassab. Technical C Programming. Prentice Hall 1989.

Marshall Cline.

C++ FAQ LITE.

http://www.parashift.com/c++-faq-lite/

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream brary

ontinuing our tour

ata Types

expressions

unctions

Arrays

inters

Arrays and Pointers

Multidimensional Arrays and arrays of pointers

tructures

Reading and Writing

Guidelines



Resources III



Bjarne Stroustrup.

A Tour of the Standard Library.

Chapter 3 of The C++ Programming Language.

http:

//public.research.att.com/~bs/3rd_tour2.pdf

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream

Continuing our tour

ata Types

Lxpressions

Statements

Arrays

. . .

Arrays and Pointers

Multidimensional

Multidimensional Arrays and arrays of pointers

ructures

Reading and Writing

Guidelines

License covering this document

Copyright © 2006 Nick Urbanik <nicku@nicku.org>
You can redistribute modified or unmodified copies of this
document provided that this copyright notice and this
permission notice are preserved on all copies under the terms
of the GNU General Public License as published by the Free
Software Foundation — either version 2 of the License or (at
your option) any later version.

C Programming for Engineers

Nick Urbanik

Who am I? Who are You?

Overview

C++ and iostream
library

ontinuing our tou

ata Types

LAPIESSIONS

Statements

.

Arrays

ointers

Arrays and Pointers

Multidimensional Arrays and arrays of

uctures

Reading and Writing iles

auidelines

