

Introduction to C++

Fundamentals of C++
Additional material can be found in
the online module

What is C++

- C++ is a high level programming language because it is written in pseudo english
- It was derived from the C programming language (developed by Kernighan and Ritchie).
- Used in developing desktop applications, embedded systems, and system applications.

High Level Languages

- A high level programming language takes care of all memory management and utilizes either a compiler or an interpreter to translate the language into machine language.
- Grammatical Rules
 - Keywords
 - Used to translate the high-level language into machine language.

Program Structure

- Header files at the beginning of the program
- All C++ programs must have the function main in them
 - int main() is called the function header
 - int specifies that a return code will be returned to the operating system that is used to indicate if the program terminated normally. The number zero has traditionally been used to indicate normal termination.
 - Specifies the starting point of the program
 - the word main must be in all lower case letters, since c++ is case sensitive

C++ Skeleton

Preprocessor directives
Constant Declarations
Class Definitions
Function Prototypes

**Global
Declaration
Section**

```
int main( ) { ----- Driver Routine (required)
  Local declarations----- Variables are declared
  program statements----- Process variables
  return 0; ----- Used by C++
}
```

Function Definitions ----- top-down design

Example Program

Preprocessor directive (#)
(Semicolon is omitted on all preprocessor directives.)

```
#include <iostream>
using namespace std;
int main(){
  cout << "Hello World " << endl;
  return 0;
}
```

system library to allow input/output

Program Statements

C++ Statements end in a semicolon i.e. ;

Display statement in C++. Sends information to the monitor

PreProcessor Directives

- All preprocessor directives begin with a # sign and no space is allowed between the pound sign and the keyword. (Helps the compiler)
- A **keyword** is a word that has a special meaning to the compiler (allows translation of the programming language into machine language).
- **using namespace std** means you will be using the standard library. Required for the majority of commonly used statements in C++.

iostream

- Is a header file to allow for console input (**cin**)(keyboard) and console output (**cout**) (sending information to the monitor).
- A library of predefined programs to allow interaction with the computer hardware

C++ Terminology: **Identifier**

- An identifier is the name used for a variable or a constant,
- or for a function,
- or for a data type, within a C++ program.

C++ Rules for Identifiers

1. An identifier must start with a letter or the underscore (A – Z, a-z, or _)
2. Can be followed by zero or more letters (A-Z, a-z), digits (0-9), or underscores.
3. Can not be a reserved word.

- **VALID**

age_of_dog taxRate98
printHeading ageOfHorse

- **NOT VALID (Why?)**

age# 98TaxRate age-Of-Cat

Remember this example

- A home mortgage authority requires a deposit on a home loan according to the following schedule:

Loan Amount	Deposit Required
Less than \$25,000	5% of the loan value
\$25,000 to \$49,999	\$1250 + 10% of the loan amount over \$25,000
\$50,000 to \$100,000	\$5000 + 25% of the loan amount over \$50,000

- Loans in excess of \$100,000 are not allowed.
- Design an algorithm that will read a loan amount and compute and print the required deposit.

The Pseudocode Algorithm

```
get loanAmount
set deposit to zero
if loanAmount > 100000 then
    display "Loans can not exceed 100000"
else if loanAmount >= 50000 then
    set deposit to 5000 + .25 * (loanAmount - 50000)
else if loanAmount >= 25000 then
    set deposit to 1250 + .1 * (loanAmount - 25000)
else
    set deposit to .05 * loanAmount
Display "Amount of deposit is ", deposit
```

What are the variables in the algorithm?

- loanAmount and deposit
- These variables will be used in a C++ program to implement this algorithm.

What is a Variable?

- A **variable** is an identifier which references a memory location.
- A variable will contain a **value that can be changed during program execution. Just like when we traced our pseudocode algorithm (i.e. playing computer).**

What is a Variable continued

- **All variables must be declared before they can be used.**
- The declaration has the following format:
data-type legalIdentifierName
Used by the compiler to make a request to the memory manager of the OS to allocate the required memory for the variable.

C++ Data Types

- A type defines a set of values and a set of operations that can be applied on those values. The set of values is called the domain for the type.
- 5 standard types
 1. void [has no values or operations. Used for generic types] --- used in advanced programming
 2. int (short for integer)
 3. char (short for character)
 4. float and double (floating point numbers, i.e. decimal)
 5. bool (short for boolean)

Integer *int* Data Types

- A whole number
- declared as **int**, **short**, or **long**
- Number of bytes allocated is dependent on the compiler. Visual Studio allocates 4 bytes for **int** or **short** declarations. The allocation is 8 bytes if the **long** declaration is used.

Character *char* Data Types

- Anything typed on the keyboard is of type ASCII (ask-key) which is an acronym for American Standard Code for Information Interchange.
- Since char data types consist of one byte, we have 2^8 possibilities, which is 256 possible values.

Floating-point Type

- Consists of 3 fundamental types
 - float, double, and long double
- Note: The visual studio compiler treats all numbers of the form, 2.4, as type double. Therefore we will declare all floating point variables as type **double**.
- Visual C++ will generate warning messages if you use type float.

Boolean *bool* Data Types

- Used for logical comparisons. Named after George Boole, a mathematician.
- Is either true or false.
- Two constants that are reserved words: **true** and **false**.
- When used in conjunction with integer data types, the bool values are converted to integer using the following rules:
 - true is converted to one**
 - false is converted to zero**
- Anything non-zero in C++ is treated as a true value

Descriptive Naming

- The names or identifiers used for variables should be descriptive, i.e. what will they contain or how will they be used, etc.
- This is a form of self-documenting code.
- For example, which is more descriptive?
 - sum** or **totalSales**

What Does a Variable Declaration Do?

A declaration tells the compiler to allocate enough memory to hold a value of this data type, and to associate the identifier with this location.

```
int    ageOfDog; // variable declaration
double taxRate98; // variable declaration
char   MiddleInitial = 'B';
```



4 bytes for TaxRate98



1 byte for MiddleInitial

Literal Constants

- A literal is an unnamed constant
- e.g.
 - '5' {type char literal}
 - 78 {type int literal}
 - a + 5 {dependent on the datatype of the identifier a}
 - "Hello World" {string literal}
 - 3.1416 {float or double literal}

From the previous pseudocode algorithm

- loanAmount and deposit
 - are the variables we used in the algorithm
 - A decision has to be made to the data type these variables should be: int, char, double, or bool?
 - Which one(s)?

Variable Initialization

- When you declare a variable, it does not contain any type of value. It just allocates storage for the identifier.
- Initialization means to assign a value to a variable.
- This can be done during compilation (compile time) or during program execution (runtime).

Pseudocode get statement and the corresponding C++ statements

- get loanAmt is translated to following C++ statements
 - `cout << "Enter the amount of the loan ";`
 - `cin >> loanAmt;`
- `cout << "Enter the amount of the loan ";` is called a prompt and causes the contents between " and " to display on the console monitor.
- `cin >> loanAmt;` means to accept whatever is typed at the keyboard and place the result into loanAmt. Note: Execution does not pause if the input buffer is not empty. Input terminates when the input data type does not match the variable.

C++ Program so far

```
#include <iostream>
using namespace std;
int main(){
    double loanAmt, deposit;
    cout << "Enter the amount of the loan ";
    cin >> loanAmt;
    // more statements ---- comments visible only to the programmer
    return 0;
} // end main
```

Output Statements

SYNTAX --- Grammar of the language

```
cout << ExprOrStringOrManipulator
      << ExprOrStringOrManipulator ...;
```

Optional

Examples of cout:

```
cout << 3 ; // displays the number 3 to the monitor
```

```
cout << "Hello World" << " hi" ;
```

```
// displays the string Hello World hi
```

If outputting more than one value, each value must be separated by the insertion operator.

Cout Continued

- String constants cannot extend beyond the physical end of line.
- The insertion operator, <<, must separate each item to display to the monitor.
- Example:

```
int able = 4, baker = 5;
cout << "The value of able is " << able
      << " baker is " << baker << endl;
```

The output is:

```
The value of able is 4 baker is 5
```

able is not an identifier when used within a string literal

baker is not an identifier when used within a string literal

Input

- Input into a C++ program from the user can be from a disk file or through the keyboard. If keyboard input is desired, we use the istream object **cin**.
- Variable **cin** is predefined to denote an input stream from the standard input device (the keyboard). **cin** stands for **C**onsole **i**nput
- Data can be entered into a C++ program through the keyboard in the form of a sequence of characters. The characters are automatically converted to the data type of the variable.

Extraction Operator (>>)

- The extraction operator >> called “get from” and requires 2 operands. The left operand is a stream object, such as **cin**. The right operand is a variable.
- Since the extraction operator requires two operands it is classified as a **binary operator**.
- Operator >> attempts to extract the next item from the input stream and store its value in the right operand variable. **The input value must match the data type of the variable.**

Input Statements

SYNTAX

cin >> Variable >> Variable...;

These examples yield the same result.

cin >> length ;

cin >> width ;

cin >> length >> width ;

cin >> length
>> width;

Note: If specifying more than one variable with cin, you must separate the variables with the extraction operator.

Optional

Input Statements

- Execution of your program will pause as long as the input buffer is empty.
- The user enters values into your program through the keyboard and by pressing the enter key.
- If the input buffer is not empty, execution will not pause.

C++ Statement similar to set

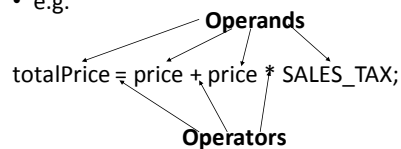
- The C++ statement used to implement the pseudocode statement set is called the assignment statement.
- The syntax of the assignment statement is:
 - variable = expression;
- Result of the expression is placed into the variable on the left hand side of the assignment operator (=).

What is an Expression?

- An expression is any combination of constants, variables, or arithmetic operators.
- The result of the expression will always be a value. The data type of the value is dependent on the make up of the expression.
- I.e. $3.5 + 3$ results in 6.5 which is type double
- I.e. $3 + 2$ results in 5 which is type int.
- bool and char results can occur and are covered in 1106.

Operators vs Operands

- An operator is a language-specific syntactical token that requires an action to be taken
- An operand receives the operators action.
- e.g.



Some C++ Operators

Precedence	Operator	Description
18	()	Parentheses
16	++ or - -	Postfix increment or decrement
15	+ or -	Unary plus or minus
13	*, /, %	Multiply, Divide, Modulus (Remainder)
12	+, -	Addition, Subtraction
2	=, *=, %=, +=, -=, /=	Assignment

Division Operator

- The result of the division operator depends on the data types of its operands.
- If one or both operands has a floating point type, the result is a floating point type. Otherwise, the result is an integer type.
- Examples

$11 / 4$ has value 2
 $11.0 / 4.0$ has value 2.75
 $11 / 4.0$ has value 2.75

operand → $11.0 / 4.0$ $11 / 4.0$ ← operand

Modulus Operator

- The modulus operator % can only be used with integer type operands and **always has an integer type** result.
- Its result is the integer type **remainder** of an integer division.
- EXAMPLE

$11 \% 4$ has value 3 because

$$\begin{array}{r} \text{R = ?} \\ 4 \overline{) 11} \end{array}$$

Evaluate the Expression

means

$$\begin{aligned}
 &7 * 10 - 5 \% 3 * 4 + 9 \\
 &(7 * 10) - 5 \% 3 * 4 + 9 \\
 &70 - 5 \% 3 * 4 + 9 \\
 &70 - (5 \% 3) * 4 + 9 \\
 &70 - 2 * 4 + 9 \\
 &70 - (2 * 4) + 9 \\
 &70 - 8 + 9 \\
 &(70 - 8) + 9 \\
 &62 + 9 \\
 &71
 \end{aligned}$$

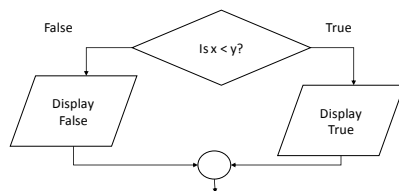
Parentheses

- Parentheses can be used to change the usual order
- Parts in () are evaluated first
- Evaluate $(7 * (10 - 5) \% 3) * 4 + 9$

$$\begin{aligned}
 &(7 * 5 \% 3) * 4 + 9 \\
 &(35 \% 3) * 4 + 9 \\
 &2 * 4 + 9 \\
 &8 + 9 \\
 &17
 \end{aligned}$$

Decision or Selection Structures

- A Decision structure is used when an alternate path of execution is desired. The flowchart representation of this structure is:



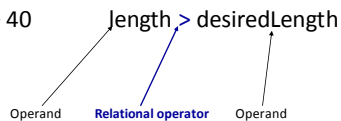
Relational Operators

- The computer can make comparisons using the standard operators below:

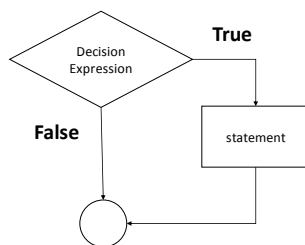
Relational Operator	Meaning	Example
<	Less Than	$x < 10$ or $x < y$
>	Greater than	<code>totalSales > minSales</code>
<=	Less than or equal to	<code>totalSales <= minSales</code>
>=	Greater than or equal to	<code>totalSales >= minSales</code>
==	Equal to	<code>totalSales == minSales</code>
!=	Not Equal to	<code>totalSales != minSales</code>

Relational Expressions

- Expressions formed between two operands and a relational operator.
- A simple relational expression consists of a relational operator connecting two variable and/or constant operands.
- Etc. `age > 40`



One-Way Selection

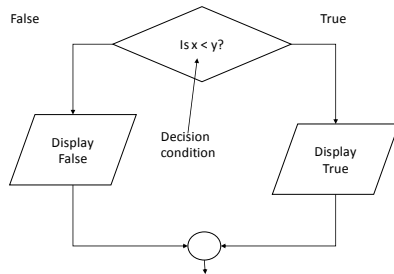


C++ Implementation

```
if(expression)
    statement1;
```

- If the expression is true, statement1 will be executed, otherwise it will be skipped.
- If more than one statement is needed after the if(expression), you must enclose them in curly braces. (i.e. A Block or a Compound Statement)

Two-Way Selection



C++ Implementation

- The two-way selection is implemented in C++ by using the if statement.
- The syntax for the if statement is:


```
if(expression)
    statement1;
else
    statement2;
```

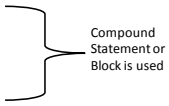
Note: A single statement. If more than one statement is required, you must use a block or a compound statement

- If the expression is true, statement1 is executed, otherwise statement2 is executed. **ONLY ONE OF THE STATEMENTS WILL BE EXECUTED.**

if-else continued

- More than one statement is required:

```
if(expression){
    statement1;
    statement2;
    statement3;
}
else
    statement4;
```



Compound Statement or Block is used

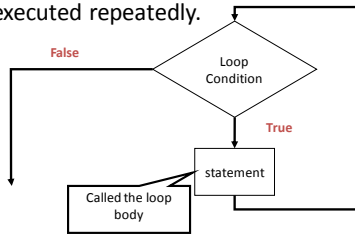
Example C++ Program

Loops often used for

- Counting all data values
- Count only special data values
- Sum up and get a total of all the data that has been input
- Keep track of previous and current values
- Searching through a list for a particular value
- Sorting a list of values
- Finding the smallest or largest value in a list.

What is a loop?

- A loop is a repetition control structure.
- It causes a single statement or block to be executed repeatedly.



PreTest vs. PostTest Loops

- A **PreTest** loop tests the loop condition first. If true, the loop body is executed until the loop condition becomes false. If the loop condition is initially false, the loop body will not be executed.
- **PostTest** loops are used when you want the loop body to execute at least once. The test of the loop condition is done after the loop body has executed at least once. The loop body will continue to execute as long as the loop condition is true.

Two General Types of Loops

counter controlled loops

repeat a specified number of times

event controlled loops

some condition within the loop body changes and this causes the repeating to stop

While Statement

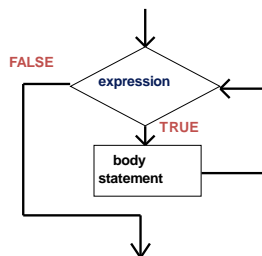
SYNTAX

```
while ( Expression )
{
    .
    .
    .
    // loop body
}
```

NOTE: Loop body can be a single statement, a null statement, or a block.

- When the expression is tested and found to be false, the loop is exited and control passes to the statement which follows the loop body.

WHILE LOOP



Event-controlled loops

• Sentinel controlled

keep processing data until a special value which is not a possible data value is entered to indicate that processing should stop.

• End-of-file controlled

keep processing data as long as there is more data in the file.

• Flag controlled

keep processing data until the value of a flag changes in the loop body.

A Sentinel-controlled loop

- Requires a “priming read”
- “Priming read” means you read one set of data before the while

Example 1

- Write a program which reads a collection of numbers and then prints the largest number in the list. Assume the numbers are terminated by the sentinel value 9999.

Example Input file: 18 23 14 3729 9999.

Example 2

- Write a program that prompts the user for miles traveled and the number of gallons used per trip and calculates the miles per gallon per trip. This process should continue until the user enters in a negative value for miles traveled.
