

Getting Started with C++

- Writing a C++ program
- Method to
 - Enter
 - Compile
 - Link

a C++ program.

Program

- A set of written instructions
- created by a programmer
- that can be executed by a computer.

Programming Language

- Machine language:
 - Instructions represented by long strings of ones and zeroes
- Assemblers:
 - Map machine instructions to human-readable and manageable mnemonics, such as ADD and MOV.
- Higher-level languages:
 - C++, Visual Basic, work with something approximating words and sentences

Interpreters, Compilers and Linkers

- Translate Higher-level languages instructions into machine language
- Interpreter: Translates a program as it reads it, turning the program instructions, or code, directly into machine language and executes it
- Compiler: Translates the code into an intermediary form called an object file.
- Linker: Turns the object file into an executable machine language program which is run for execution

Writing source code or C++ program

- Built-in text editor
- Commercial text editor
- Word processor
- Commercial word processors, such as WordPerfect, Word
- Editors EMACS and vi
- Windows Notepad
- DOS Edit command
- Visual C++ editor

Compiling the Source Code

- Borland C++ compiler: `bcc <filename>`
- Borland C++ for Windows compiler:
`bcc <filename>`
- Borland Turbo C++ compiler:
`tc <filename>`
- Microsoft compilers: `cl <filename>`
- Visual C++ compiler

Library, Functions and Classes

- Library
 - Collection of files linkable with source files. These files are supplied with compiler. All C++ compilers have a library of useful functions (or procedures) and classes that can be included in the source program.
- Function
 - A block of code that performs a service, such as adding two numbers or printing to the screen.
- Class
 - Collection of data and related functions.

Programming

- Procedural or Structured programming
- Object-Oriented Programming

Procedural or Structured Programming

- A program consists of a set of tasks also called procedures or functions
- Complex tasks are broken down into a set of smaller component tasks until tasks are
 - sufficiently small and
 - self-contained

Procedure or Function

- A set of specific instructions
- Executed one after the other.
- In Structured or Procedural Programming
 - the data is separate from the procedures.

Compute the average salary of the employees of a company

Break the problem down into these subtasks:

- 1. Find out what each person earns.
- 2. Count how many people you have.
- 3. Total all the salaries.
- 4. Divide the total salary by the number of people you have.

Breaking Down Task 1, 2 and 3 into Sub-Tasks

Tanvir Mustafy
Lecturer, MIST

- 1. Get each employee's record.
- 2. Access the salary of current record.
- 3. Add the salary to the running total.
- 4. Get the next employee's record.

Obtaining each employee's record

The above task is broken down into the following sub-tasks:

- 1. Open the file of employees.
- 2. Go to the correct record.
- 3. Read the data from disk.

Object-oriented Programming

- Treat data and the procedures that act upon the data as a single OBJECT
- An OBJECT is a self-contained entity with
 - an identity and certain characteristics of its own, which are represented by data
 - procedures or functions enabling it to perform tasks

Four Pillars of Object-Oriented Programming

Tanvir Mustafy
Lecturer, MIST

- encapsulation
- data hiding
- inheritance
- polymorphism

C++ fully supports object-oriented programming

Pillars of Object Oriented Programming

- Encapsulation
 - being a self contained unit
- Data hiding
 - an object can be used without knowing about its internal data members

Encapsulation and Data Hiding

- C++ supports encapsulation and data hiding through
- the creation of user-defined types, called classes.
- a well-defined class acts as a
 - fully encapsulated entity--it is used as a whole unit
 - the actual inner workings of the class are hidden
 - users of a well-defined class do not need to know how the class works
 - they just need to know how to use it

Inheritance and Polymorphism

- Inheritance
 - a new object type, which is an extension of an existing type can be declared
 - this new subclass is said to derive from the existing type and is sometimes called a derived type
 - C++ supports reuse through inheritance.
- Polymorphism
 - refers to the same name taking many forms
 - C++ supports the idea that different objects do "the right thing" through
 - function polymorphism and class polymorphism

Evolution of C++

- C was extended to create C++ by Bjarne Stroustrup
- Features needed to facilitate object-oriented programming was provided in C++
- C++ is the predominant language for commercial software development
- C++ is a superset of C. The leap from C to C++ is very significant
- Any legal C program is a legal C++ program.
- To get the full benefit of C++, programmers need to learn a new way of conceptualizing and solving programming problems

The ANSI Standard

- American National Standards Institute (ANSI),
- Create an international standard for C++.
- ANSI standard attempts to ensure that C++ is portable
- Code written for Microsoft's compiler will compile without errors, using a compiler from any other vendor.

HELLO.CPP

- 1: #include <iostream.h>
- 2:
- 3: int main()
- 4: {
- 5: cout << "Hello World!\n";
- 6: return 0;
- 7: }
- Hello World!

Compiler and Compiler Errors

- Compile Errors
 - Any error in syntax or grammar of C++.
- Compilers generally indicate
 - what mistake you made and
 - at which line and location of the program the mistake was made.

Demonstrating Compiler Error

- 1: `#include <iostream.h>`
- 2:
- 3: `int main()`
- 4: `{`
- 5: `cout << "Hello World!\n";`
- Hello.cpp, line 5: Compound statement missing terminating `}` in function main

HELP.CPP demonstrates comments

Tanvir Mustafy
Lecturer, MIST

- 1: #include <iostream.h>
- 2:
- 3: int main()
- 4: {
- 5: /* this is a comment and it extends
- 6: until the closing star-slash comment mark */
- 7: cout << "Hello World!\n";
- 8: // double slash comments can be alone on a line
- 9: /* as can slash-star comments */
- 10: return 0;}
- Hello World!

Comments at top of Hello.cpp

Tanvir Mustafy
Lecturer, MIST

- ```
/*

Program: Hello World
File: Hello.cpp
Function: Main (complete program listing in this file)
Description: Prints words "Hello world" to screen
Author: Jesse Liberty (jl)
Environment: Turbo C++ version 4, 486/66 32mb RAM,
 Windows 3.1
DOS 6.0. EasyWin module
Notes: Introductory, sample program
Revisions: 1.00 10/1/94 (jl) First release
 1.01 10/2/94 (jl) Capitalized "World"

*****/
```

# Using cout.

- 1: `#include <iostream.h>`
- 2: `int main()`
- 3: `{ cout << "Hello there.\n";`
- 4: `cout << "Here is 5: " << 5 << "\n";`
- 5: `cout << "The manipulator endl writes a new line to`  
`the screen." << endl;`
- 6: `cout << "Here is a very big number:\t" << 70000 <<`  
`endl;`
- 7: `cout << "Here is the sum of 8 and 5:\t" << 8+5 <<`  
`endl;`
- 8: `cout << "Here's a fraction:\t\t" << (float) 5/8 <<`  
`endl;`
- 9: `cout << "And a very very big number:\t" <<`  
`(double) 7000 * 7000 << endl;`
- 10: `cout << "Don't forget to replace Jesse Liberty with`  
`your name...\n";`
- 11: `return 0;}`

# Output using cout

Tanvir Mustafy  
Lecturer, MIST

- Hello there.
- Here is 5: 5
- The manipulator endl writes a new line to the screen.
- Here is a very big number: 70000
- Here is the sum of 8 and 5: 13
- Here's a fraction: 0.625
- And a very very big number: 4.9e+07
- Don't forget to replace Jesse Liberty with your name...
- Jesse Liberty is a C++ programmer!

# Demonstrating call to a function

- 1: `#include <iostream.h>`
- 2: `// function Demonstration Function`
- 3: `// prints out a useful message`
- 4: `void DemonstrationFunction()`
- 5: `{ cout << "In Demonstration Function\n"; }`
- 6: `// function main - prints out a message, then`
- 7: `// calls DemonstrationFunction, then prints`
- 8: `// out a second message.`
- 9: `int main() { cout << "In main\n" ;`
- 10: `DemonstrationFunction();`
- 11: `cout << "Back in main\n";`
- 12: `return 0; }`

# Output for Function call

- In main
- In Demonstration Function
- Back in main

# FUNC.CPP

Tanvir Mustafy  
Lecturer, MIST

- 1: #include <iostream.h>
- 2: int Add (int x, int y) {4:
- 3: cout << "In Add(), received " << x << " and " << y << "\n";
- 4: return (x+y); }
- 5: int main() { cout << "I'm in main()!\n";
- 6: int a, b, c;
- 7: cout << "Enter two numbers: ";
- 8: cin >> a;
- 9: cin >> b;
- 10: cout << "\nCalling Add()\n";
- 11: c=Add(a,b);
- 12: cout << "\nBack in main().\n";
- 13: cout << "c was set to " << c;
- 14: cout << "\nExiting...\n\n";
- 15: return 0; }

# Output of FUNC.CPP

Tanvir Mustafy  
Lecturer, MIST

- I'm in main()!
- Enter two numbers: 3 5
- Calling Add()
- In Add(), received 3 and 5
- Back in main().
- c was set to 8
- Exiting...