LECTURE 1

Instructor Info

- Dr. Kirti Seth
  - Ph. D.
    - Computer Science and Engineering
    - India
  - M.Tech(CSE)
    - India
  - MSc
    - Computer Science
    - India
- Contact
  - IUT Office # 402-7
  - Email: k.seth@inha.uz

Course Objectives:

- To appreciate the need for a programming language.
- To introduce the concept and usability of the structured programming.
- To develop proficiency in making useful software using the C++ language.
- Analyze written problem specifications and divide those specifications into logical modules.
- Develop and document the design of a program using flowcharts.
- Develop and document the design of a program using pseudo-code.
- Convert the designs into structured programs using high-level language, i.e. C++.

Course Info

- **Text Book:**
  - C++ How to program by Dietel & Dietel, 3rd Edition
- **Suggested Reference:**
  - Object Oriented Programming in C++ by Robert Lafore, 3rd Edition

WEEK 1: Introduction

- What is a Computer and what are computer languages?
- Machine Languages, Assembly Languages, and High-level Languages
- History of C and C++
- C++ Standard Library
- Translators: Compiler, Interpreter, Assembler
- Algorithms, Pseudo code
- Structured Programming
- Basics of a Typical C++ Environment

What is computer?

- **Computer**
  - A device capable of performing computations and making logical decisions
  - A machine that manipulates data according to a list of instructions.
  - A programmable device that can store, retrieve, and process data.
- **Computer programs**
  - Sets of instructions that control a computer’s processing of data
- **Hardware**
  - Physical part of the computer
  - Various devices comprising a computer
    - Examples: keyboard, screen, mouse, disks, memory, CD-ROM, and processing units
- **Software**
  - A collection of computer programs, procedures and documentation that perform some tasks on a computer system
  - Programs that run a computer

Computer organization

There are Six logical units in every computer:

- Input unit
  - Obtains information (data and computer programs) from input devices (keyboard, mouse)
- Output unit
  - Outputs information to output device (screen, printer) or to control other devices.
- Memory unit
  - Rapid access, low capacity, stores input information
- Arithmetic and logic unit (ALU)
  - Performs arithmetic calculations and logic decisions
- Central processing unit (CPU)
  - Supervises and coordinates the other sections of the computer
- Secondary storage unit
  - Cheap, long-term, high-capacity storage, stores inactive programs

Computer languages

- Computer languages are divided into three types.
  - Machine languages
    - Set of Instruction executed directly by a computer's CPU
    - Machine language is machine dependent.
    - Strings of numbers giving machine specific instructions
    - Example:

```
+1300042774
+1400593419
+1200274027
```

  - Assembly languages
    - English-like abbreviations representing elementary computer operations (translated via assemblers)
    - Example:

```
LOAD BASEPAY
 ADD OVERPAY
 STORE GROSSPAY
```

Translator programs called **assembler** were developed to convert assembly language programs to machine language programs at computer speed.

Computer languages

- **High-level languages**
  - Similar to everyday English, use mathematical notations (translated via compilers)
  - Example:

```
grossPay = basePay + overTimePay
```

- C, C++ are the most widely used high level languages. Some other examples are
- FORTRAN (formula translator)

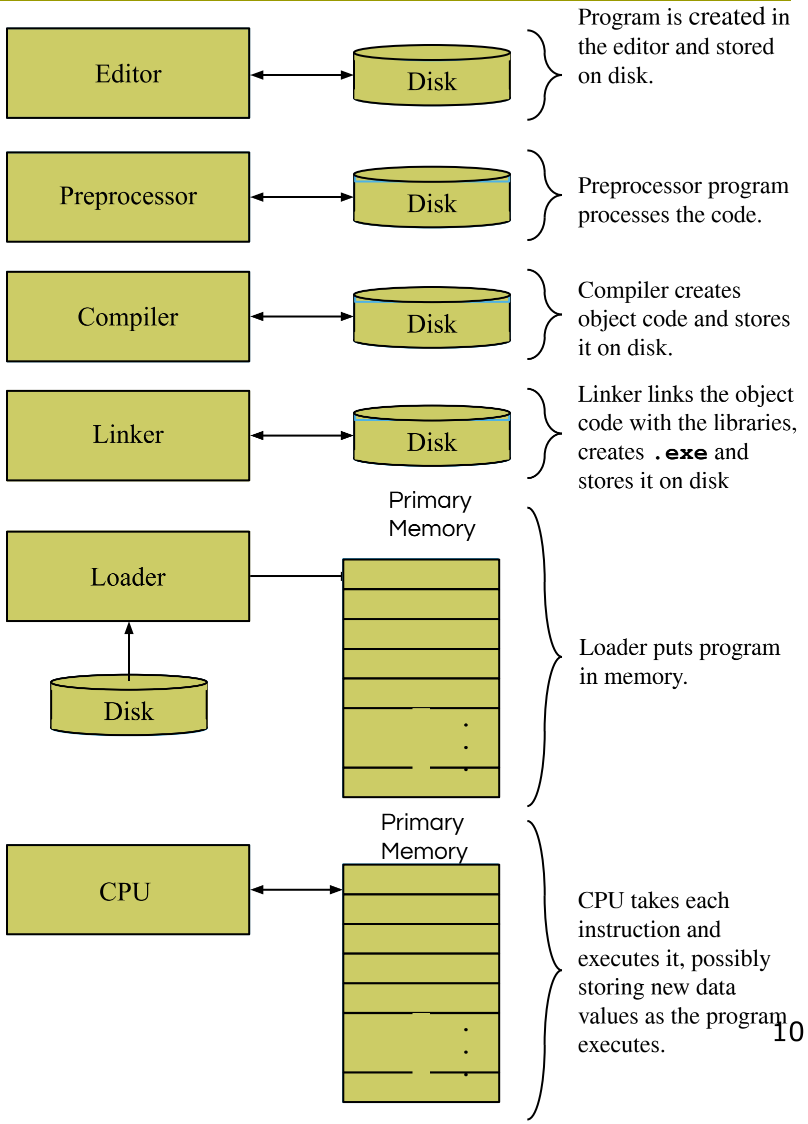
  Used in scientific and engineering applications
- COBOL (common business oriented language)

  Used to manipulate large amounts of data
- Pascal

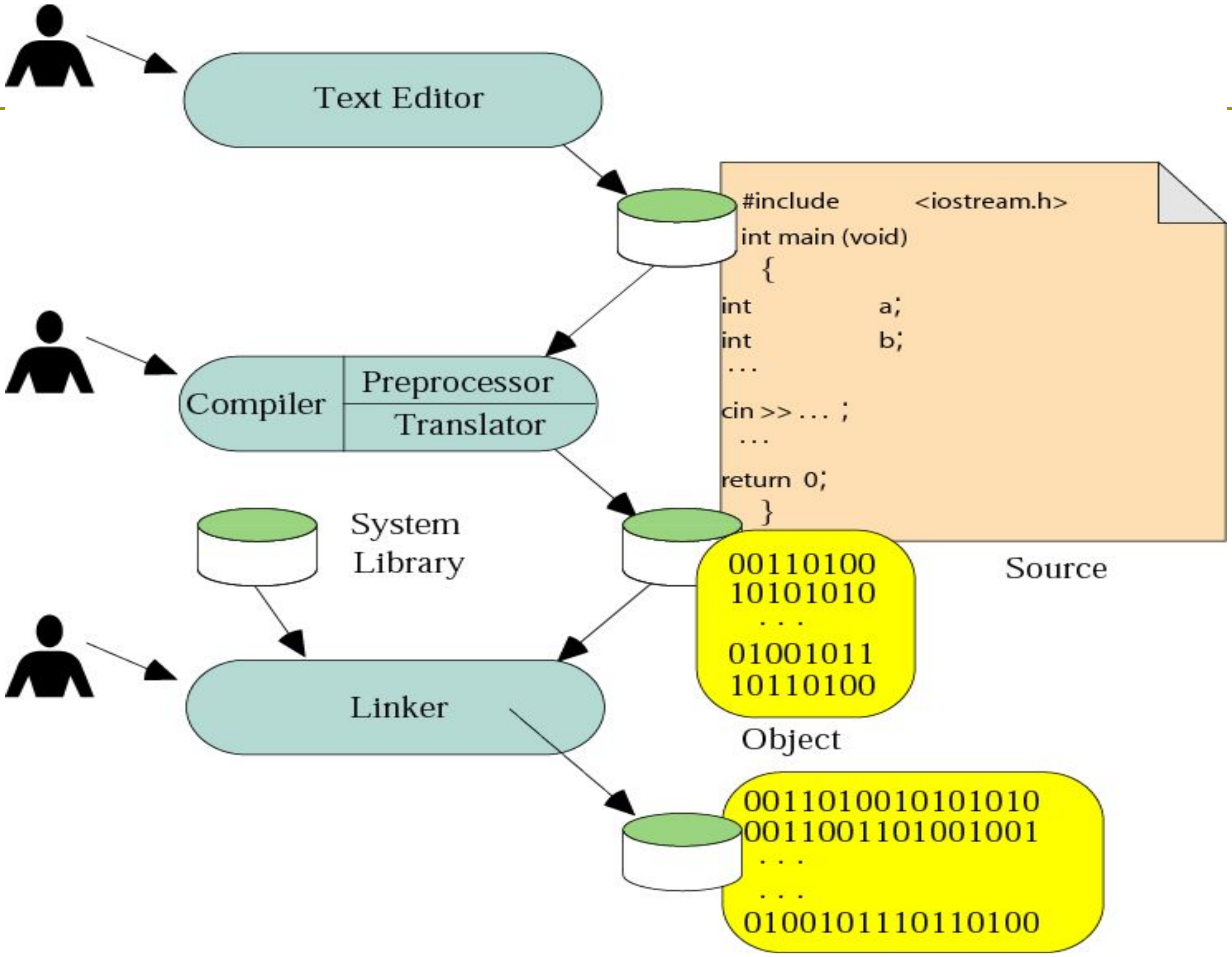
  Used to teach structured programming
- Translator programs called **Compilers** converts high-level language programs into machine language

Basics of a typical C++ environment

- Phases of C++ Programs to be executed
  - Edit
  - Preprocess
  - Compile
  - Link
  - Load
  - Execute



Editor
Disk
Program is created in the editor and stored on disk.
Preprocessor
Disk
Preprocessor program processes the code.
Compiler
Disk
Compiler creates object code and stores it on disk.
Linker
Disk
Linker links the object code with the libraries, creates .exe and stores it on disk
Primary Memory
Loader
Disk
Loader puts program in memory.
Primary Memory
CPU
CPU takes each instruction and executes it, possibly storing new data values as the program executes.



```
#include        <iostream.h>
int main (void)
  {
int             a;
int             b;
 . . .
cin >> . . . ;
  . . .
return  0;
  }
```

```
00110100
10101010
  . . .
01001011
10110100
```

```
0011010010101010
0011001101001001
  . . .
  . . .
0100101110110100
```

Program organization

- Program statement
  - Definition
  - Declaration
  - Action
- Executable unit
  - Named set of program statements
  - Different languages refer to executable units by different names
    - Subroutine: Fortran and Basic
    - Procedure: Pascal
    - Function : C++

C++ programming

- C++ program
  - Collection of definitions, declarations and functions
  - Collection can span multiple files
- Advantages
  - Structured into small understandable units
  - Complexity is reduced
  - Overall program size decreases

Programming and Problem Solving

- **Pseudo code**
  - Artificial, informal language used to develop algorithms
  - Similar to everyday English

- **Not executed on computers**
  - Used to think out program before coding
    - Easy to convert into C++ program
  - Only executable statements
    - No need to declare variables

Programming and Problem Solving

- **Algorithm**
  - A sequence of precise instructions which leads to a solution
- **Program**
  - An algorithm expressed in a language the computer can understand

Program Design

- Programming is a creative process
- **Program Design Process**
  - Problem Solving Phase
    - Result is an algorithm that solves the problem
  - Implementation Phase
    - Result is the algorithm translated into a programming language

Problem Solving Phase

- Be certain the task is completely specified
  - What is the input?
  - What information is in the output?
  - How is the output organized?
- Develop the algorithm before implementation
  - Experience shows this saves time in getting your program to run.
  - Test the algorithm for correctness

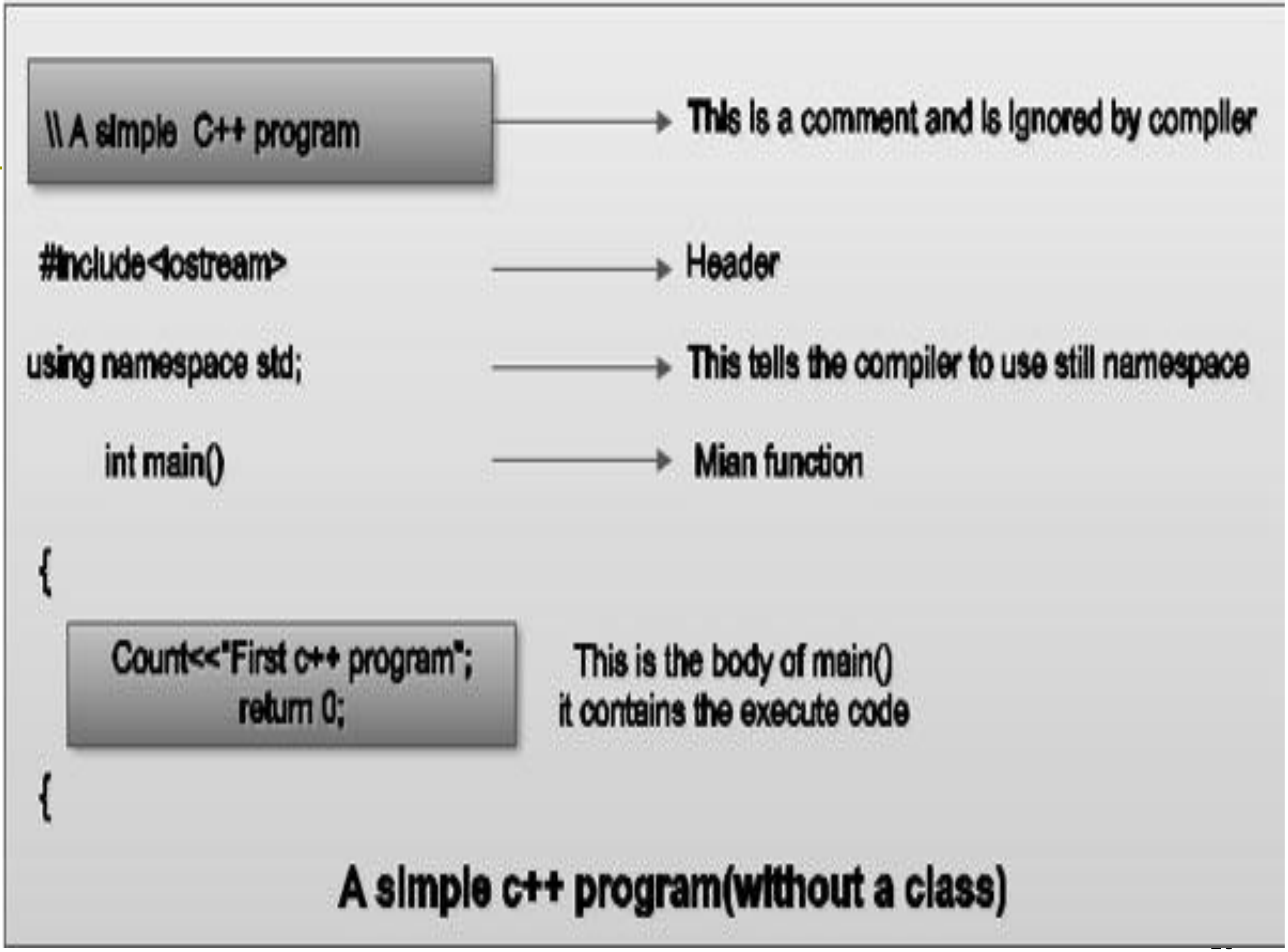
Implementation Phase

- Translate the algorithm into a programming language
  - Easier as you gain experience with the language
- Compile the source code
  - Locates errors in using the programming language
- Run the program on sample data
  - Verify correctness of results
- Results may require modification of the algorithm and program

Structure of C++ Program

hash sign

`#include <iostream>`	→ Preprocessor directives
`#include <conio.h>`	
`using namespace std;`	→ Namespace library
`int main()`	→ Main function header
`{` brace	→ Open the block
	→ Write declarations and statements
`}` brace	→ Close the block



```
\\ A simple  C++ program
#include<iostream>
using namespace std;
     int main()
{
   Count<<"First c++ program";
        return 0;
{
```

This is a comment and is ignored by compiler

Header

This tells the compiler to use still namespace

Mian function

This is the body of main()
it contains the execute code

A simple c++ program(without a class)

C++ Programming

Simple program to print a line of text.

```
1    // A first program in
2
3
4
5
6         cou
7
8         return 0;      // indicate tl
successfully
9    }
```

Comments

etween **/*** and ***/** or following a **//**.

program readability and do not cause the to perform any action.

return is a way to exit a function from a function.

return 0, in this case, means that the program terminated normally. It is one of the several means used to exit a function

preprocessor directive

preprocessor.

are preprocessor directives.

ream> tells the preprocessor

The left brace ,{, line 5 must begin the body of function and the corresponding right brace ,}, line 9 must end the body of each function.

one or more functions, one of which must be **main**

Prints the *string* of characters contained between the quotation marks.

The entire line, including **cout**, the **<<** *operator*, the *string* **"Welcome to C++!\n"** and the *semicolon* (**;**), is called a *statement*.
All statements must end with a semicolon.

C++ Programming

- **cout**
  - Standard output stream object
  - “Connected” to the screen

- **<<**
  - Stream insertion operator
  - Value to the right of the operator (right operand) inserted into output stream (which is connected to the screen)
  - **`cout << “Welcome to C++!\n”;`**

C++ Programming

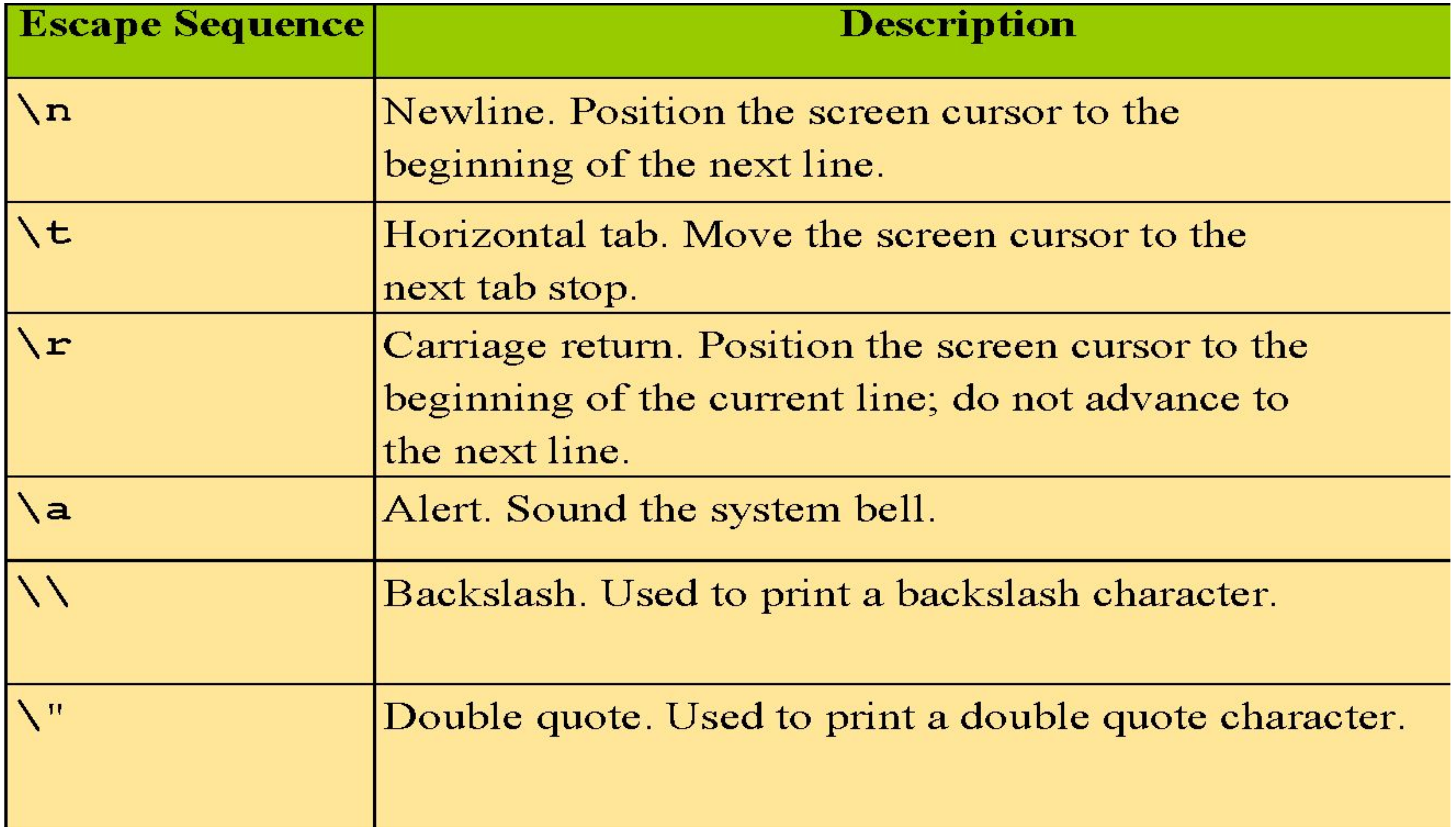
```cpp
// an example to observe using statement
// program to display greeting
#include <iostream.h>

int main()
{
   cout << "Hello world\n";

   return 0;      // indicate that program ended successfully
}
```

Escape Character

- Indicates that a “special” character is to be output



Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

C++ Programming

- There are multiple ways to print text. Following are some more examples.

```
//observing the use of \n
// Printing a line with multiple statements
#include <iostream.h>

int main()
{
   cout << "Welcome ";
   cout << "to C++!\n";

   return 0;   // indicate that program ended successfully
}
```

The output would be as bellow

```
Welcome to C++!
```

Unless new line `'\n'` is specified, the text continues on the same line.

C++ Programming

```
// printing multiple lines with a single statement
// Printing multiple lines with a single statement
#include <iostream.h>

int main()
{
   cout << "Welcome\nto\n\nC++!\n";

   return 0;   // indicate that program ended successfully
}
```

```
Welcome
To
C++!
```

Multiple lines can be printed with one statement.

Testing and Debugging

- **Bug**
  - A mistake in a program
- **Debugging**
  - Eliminating mistakes in programs
  - Term used when a moth caused a failed relay on the Harvard Mark 1 computer. Grace Hopper and other programmers taped the moth in logbook stating:
    "First actual case of a bug being found."

Program Errors

- **Syntax errors**
  - Violation of the grammar rules of the language
  - Discovered by the compiler
    - Error messages may not always show correct location of errors
- **Run-time errors**
  - Error conditions detected by the computer at run-time
- **Logic errors**
  - Errors in the program’s algorithm
  - Most difficult to diagnose
  - Computer does not recognize an error

Structured Programming

- Structured Programming is a programming paradigm aimed at improving the clarity, quality and development time of a computer program by making extensive use of subroutines (Functions), looping (e.g. for,while) etc..

C++ Standard Library

- C++ Programs consist of pieces called classes and functions. You can program each piece yourself, but most C++ programmer take advantages of the rich collections of classes and functions in the C++ standard Library.

Our Focus (Two part of learning C++)

- The first is learning C++ language itself
- The second is learning how to use the classes and functions in the C++ standard Library.