Programming Fundamentals- I

Rao Muhammad Umer
Lecturer,
Web: raoumer.github.io
Department of Computer Science & IT,
The University of Lahore.

Administrative Stuff

- Course related stuff is available on following link:

  https://piazza.com/uol.edu.pk/fall2016/cs1012/home

Comparison of C & C++

C++ Code:

```
#include <iostream>
int main()
{
 std::cout << "Welcome to C++!\n";
getch();
 return 0;
 }
```

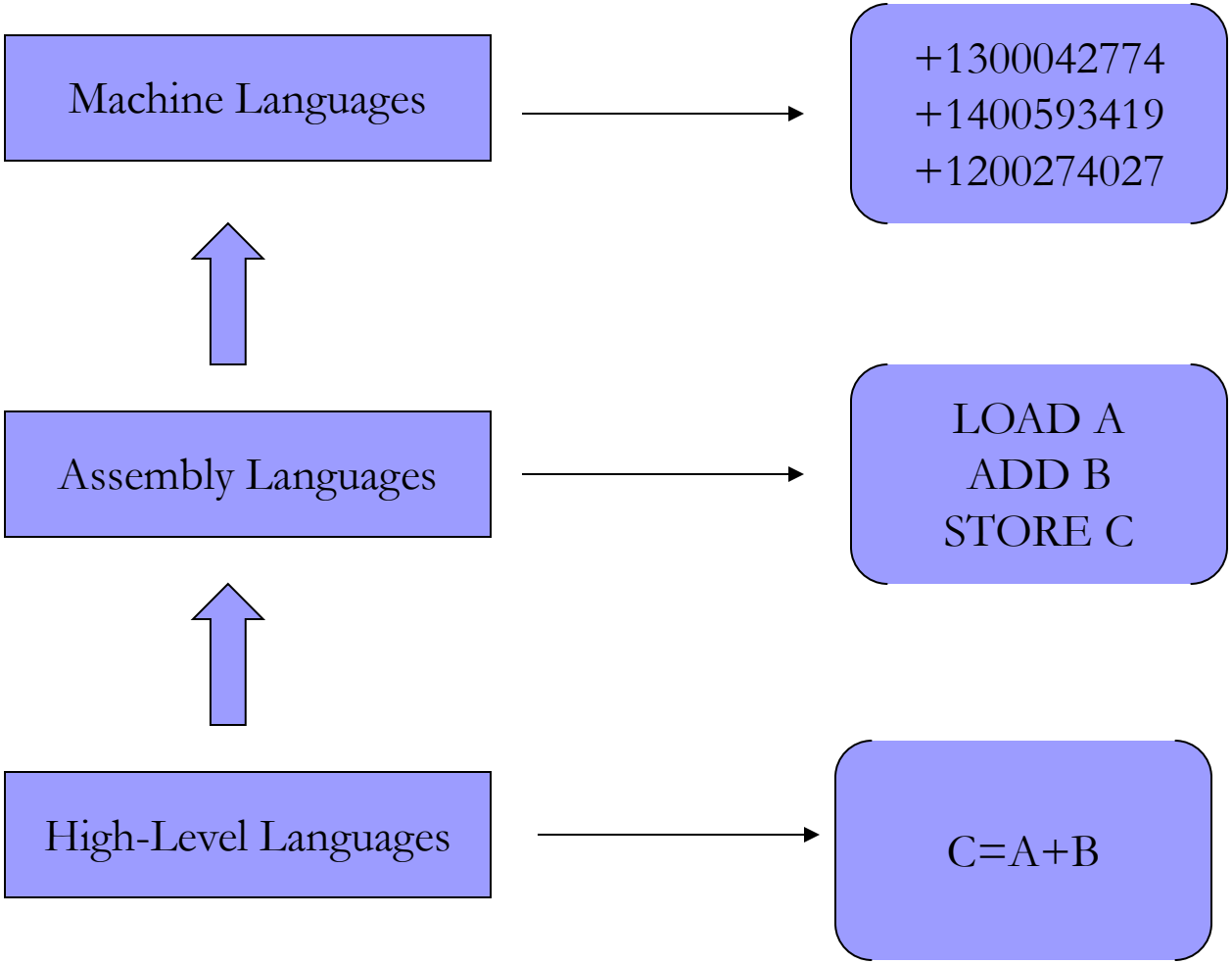
C Code:

```
#include <stdio.h>
int  main ( )
{
printf  ("Hello World!\n”);
getchar();
return 0;
}
```

Text Book

C++ How to Program

By: Deitel & Deitel (8^{th} or higher edition)


Machine Languages
+1300042774
+1400593419
+1200274027
Assembly Languages
LOAD A
ADD B
STORE C
High-Level Languages
C=A+B

Assembler

- Instructions written in assembly language must be translated to machine language instructions :
  - ❖ Assembler does this
- One to one translation : One AL instruction is mapped to one ML instruction.
- AL instructions are CPU specific.

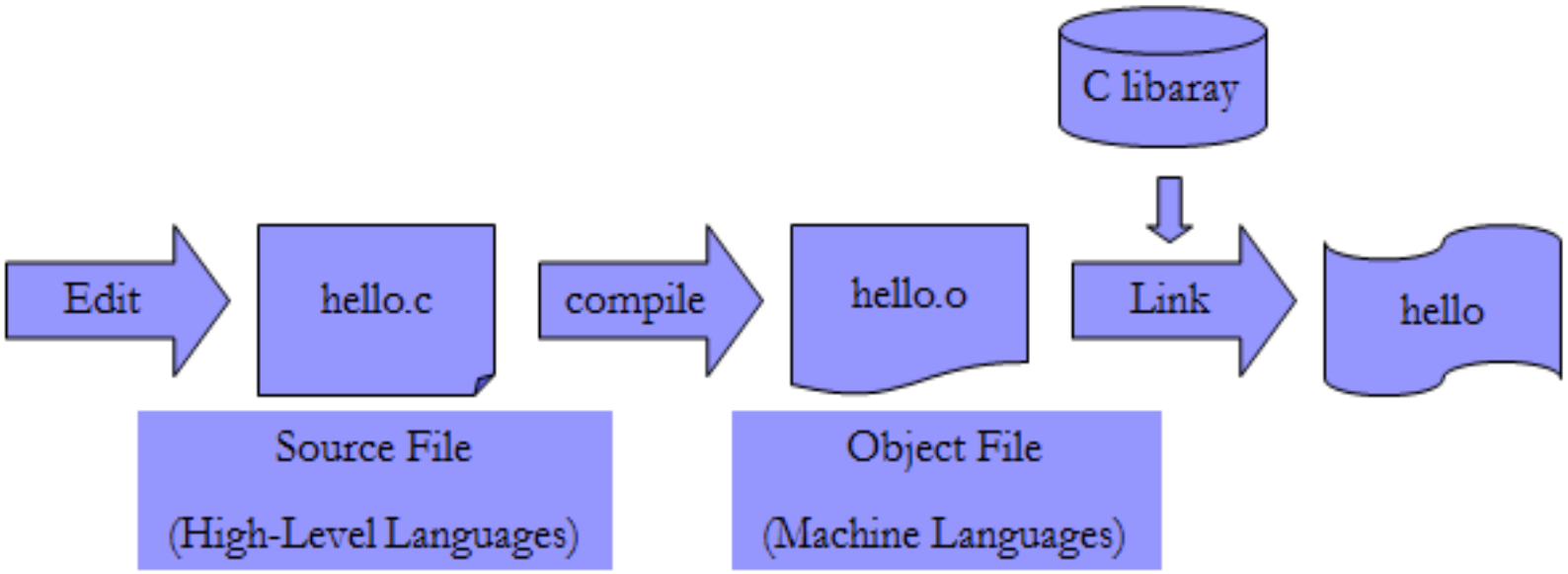
Compiler

- Instructions written in high-level language also must be translated to machine language instructions :
  - Compiler does this
- Generally one to many translation : One HL instruction is mapped to many ML instruction.
- HL instructions are not CPU specific but compiler is.

Interpreter

- An interpreter translates high-level instructions into an intermediate form, which it then executes. In contrast, a compiler translates high-level instructions directly into machine language.
- Compiled programs generally run faster than interpreted programs.
- The advantage of an interpreter, however, is that it does not need to go through the compilation stage during which machine instructions are generated. This process can be time-consuming if the program is long. The interpreter, on the other hand, can immediately execute high-level programs. For this reason, interpreters are sometimes used during the development of a program, when a programmer wants to add small sections at a time and test them quickly.

Creating Programs



Control Statements

Conditional Tasks

- **if** it is the condition, then I will do **task A**

Real Life Examples:

- **if** it is the condition, then I will do **task A**, **else**(i.e. otherwise), I will do **task B**.

Real Life Examples:

- **if** it is the condition, then I will do **task A**, else if it is the condition then I will do task B, **else** I will do **task C**.

Real Life Examples:

if statements

Example

```
#include <iostream>
using namespace std;
int main()
{
	int age;
	cout << "Please enter your age in years\n";
	cin >> age;
	if (age<=12)
		cout << "Please go to Child Specialist in Room 10\n";
	cout << “Allah Hafiz";
	getch();
	return 0;
}
```

The if Selection Structure

Selection structure:

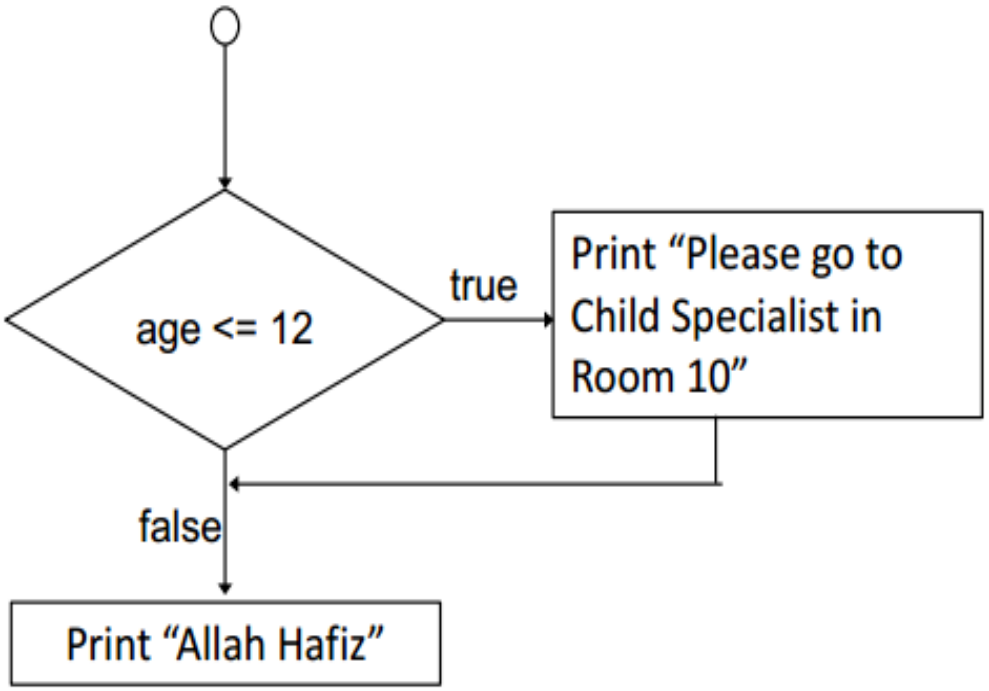
- Used to choose among alternative courses of action

```
If (age<=12)
cout << "Please go to Child Specialist in Room 10\n";
```

- If condition **true** cout statement executed and program goes on to next statement
- If **false**, print statement is ignored and the program goes onto the next statement
- Indenting makes programs easier to read

The if Selection Structure

if structure is a single-entry/single-exit structure



Diamond symbol (decision symbol)
Indicates decision is to be made contains an expression that can be `true` or `false`
Test the condition, follow appropriate path

Example

```
#include<stdio.h>
#include <stdafx.h>
int main()
{
	int age;
	cout << "Please enter your age in years\n";
	cin >> age;
	if (age<=12)
	{
		cout << "Please go to Child Specialist in Room 10\n";
		cout << " Fee is Rupees 400/=\n";
	}
	cout << "Allah Hafiz";
	getch();
	return 0;
}
```

The if Statement

- Form 1:

```
if (expression)
statement1;
next statement;
```

Example

```
#include<stdio.h>
#include <stdafx.h>
int main()
{
      int age;
      cout << "Please enter your age in years\n";
      cin >> age;
      if (age<=12)
            cout << "Please go to Pediatrics in Room 10\n\n";
      if (age>12)
            cout << "Please go to Medical Specialist in Room 15\n";
      cout << “Allah Hafiz";
      getch();
      return 0;
}
```

if else statements

Example

```
#include<stdio.h>
#include <stdafx.h>
int main()
{
	int age;
	cout << "Please enter your age in years\n";
	cin >> age;
	if (age<=12)
		cout << "Please go to Pediatrics in Room 10\n\n";
	else
		cout << "Please go to Medical Specialist in Room 15\n";
	cout << “Allah Hafiz";
	getch();
	return 0;
}
```

The if Statement

- **Form 1:**

```
if (expression)
statement1;
next statement;
```

- **Form 2:**

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

The if/else Selection Structure

- **if**
  - Only performs an action if the condition is true
- **if/else**
  - Specifies an action to be performed both when the condition is true and when it is false
- Once again

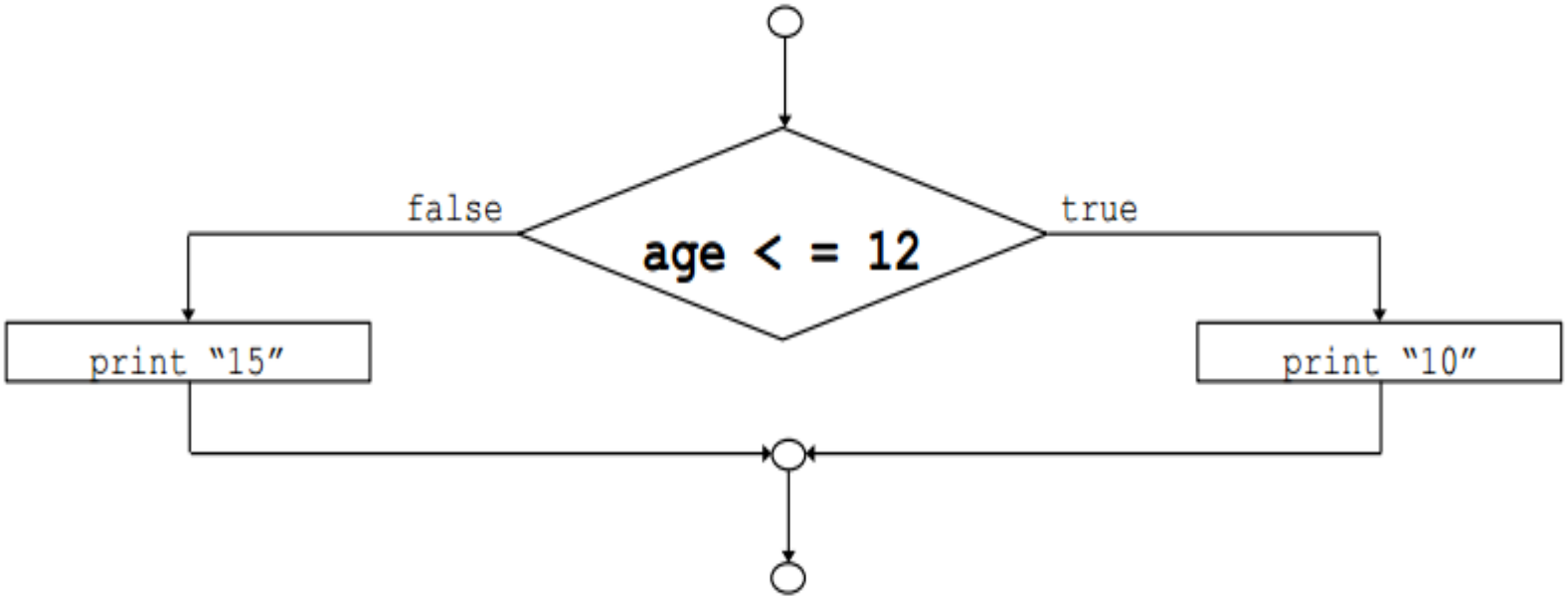
```
if (age<=12)
        cout << "Please go to Pediatrics in Room 10\n\n";

else
        cout << "Please go to Med. Spec. in Room 15\n";
```

  - Note spacing/indentation conventions

The if/else Selection Structure

Flow chart of the if/else selection structure



Example

```
#include<stdio.h>
#include <stdafx.h>
int main()
{
	int age;
	cout << "Please enter your age in years\n";
	cin >> age;
	if (age<=12)
		cout << "Please go to Child Specialist in Room 10, Fee is Rupees 400/=\n“;
	if (age > 12 && age < 60 )
		cout << "Please go to Medical Specialist in Room 15, Fee is Rupees 400/=\n";
	if (age >= 60)
		cout << "Please go to Medical Specialist in Room 19, Fee is Rupees 200/=\n";
	cout << “Allah Hafiz";
	getch();
	return 0;
}
```

if else if statements

Example

```
#include<stdio.h>
#include <stdafx.h>
int main()
{
	int age;
	cout << "Please enter your age in years\n";
	cin >> age;
	if (age<=12)
		cout << "Please go to Child Specialist in Room 10, Fee is Rupees 400/=\n“;
	else if (age > 12 && age < 60 )
		cout << "Please go to Medical Specialist in Room 15, Fee is Rupees 400/=\n";
	else (age >= 60)
		cout << "Please go to Medical Specialist in Room 19, Fee is Rupees 200/=\n";
	cout << “Allah Hafiz";
	getch();
	return 0;
}
```

The if/else Selection Structure

- **Pseudocode** for a nested **`if/else`** structure

```
If student's grade is greater than or equal to 90
  Print "A"
else
  If student's grade is greater than or equal to 80
     Print "B"
  else
     If student's grade is greater than or equal to 70
        Print "C"
     else
        If student's grade is greater than or equal to 60
           Print "D"
        else
           Print "F"
```

Example

```
#include <iostream>
int main ()
{
	int x, y;
	cout << "\nInputan integer value for x: ";
	cin >> x;
	cout << "\nInputan integer value for y: ";
	cin >> y;
	if (x == y)
		cout << "x is equal to y\n";
	else if (x > y)
		cout << "x is greater than y\n";
	else
		cout << "x is smaller than y\n";
	return 0;
}
```

Sequential execution

- Sequential execution
  - Statements executed one after the other in the **order written**
  - Sequence structures: Built into C/C++.
  - Programs executed sequentially **by default**

Control Structures

- **Transfer of control**
  - When the next statement executed is not the next one in sequence
- Selection structures: C/C++ has three types:

  if, if/else, and switch
- Repetition structures: C/C++ has three types:

  while, do/while and for (Later)

Conditional Operator

Conditional Operator

```
if ( a < b)
        c = a + 5;
else
        c = b + 8;
// We can do this in compact form as
c = a < b ? a + 5 : b + 8; //Ternary conditional operator (?:)
```

Evaluate first expression. If true, evaluate second, otherwise evaluate third.

Conditional Operator

- **Ternary conditional operator (?:)**
  - Takes **three** arguments (condition, value if **true**, value if **false**)
  - Our pseudo code could be written:

```
grade >= 60 ? cout << “Passed\n”  :
cout << “Failed\n” ;
```

Conditional Operator

- The conditional operator essentially allows you to embed an **"if"** statement into an expression
- Generic Form

```
exp1 ? exp2 : exp3
```

if exp1 is true
value is exp2
(exp3 is not evaluated)

if exp1 is false,
value is exp3
(exp2 is not evaluated)

Conditional Operator

- Example:

```
z = (x > y) ? x : y;
```

  - This is equivalent to:

```
if (x > y)
        z = x;
else
        z = y;
```

Relational Operators

Relational Operators

- Relational operators allow you to compare variables.
  - They return a 1 value for true and a 0 for false.

Operator	Symbol	Example
Equals	==	x == y NOT x = y
Greater than	>	x > y
Less than	<	x < y
Greater/equals	>=	x >= y
Less than/equals	<=	x <= y
Not equal	!=	x != y

Example

```
#include<iostream>
using namesapce std;
void main()
{
	int age;
	cout << "Please enter your age in years\n";
	cin >> age;
	if (age<=12)
		cout << "Please go to Child Specialist in Room 10, Fee is Rupees 400/=\n“;
	else if (age > 12 && age < 60 )
		cout << "Please go to Medical Specialist in Room 15, Fee is Rupees 400/=\n";
	else (age >= 60)
		cout << "Please go to Medical Specialist in Room 19, Fee is Rupees 200/=\n";
	cout << “Allah Hafiz";
	getch();
}
```

Logical Operators

Logical Operators

- **&&** **AND**
- **||** **OR**
- **!** **NOT**

Logical Operators

- **&& (logical AND)**
  - Returns **`true`** if both conditions are **`true`**
- **|| (logical OR)**
  - Returns **`true`** if either of its conditions are **`true`**
- **! (logical NOT, logical negation)**
  - Reverses the truth/falsity of its condition
  - Unary operator, has one operand
- **Useful as conditions in loops**

Expression	Result
`true && false`	`false`
`true || false`	`true`
`!false`	`true`

Be careful about Equality (==) and Assignment (=) Operators

- **Dangerous error**
  - Does not ordinarily cause syntax errors

    if (x == 4)

    cout << “You are happy\n";
    - Checks value of x, if it is 4then it prints You are happy
  - Example, replacing ==with =:

    if (x = 4)

    cout << “You are happy\n";
    - This always prints You are happy
    - 4is nonzero, so expression is true.
    - Logic error, not a syntax error

      if (x = 0)

      cout << “You are happy\n";

      What’s output?

Operator Precedence

Operator	Precedence level
( )	1
~, ++, --, unary -	2
*, /, %	3
+, -	4
<<, >>	5
<, <=, >, >=	6
==, !=	7
&	8
^	9
|	10
&&	11
||	12
=, +=, -=, etc.	14

```
#include <iostream>
using namespace std;
int main()
{
    int day;
    cout << "The day number 1 means Monday\n";
    cout << "Please enter the number of day. \n The number must be any
    integer value from 1 to 7\n";
    cin >> day;
    if (day == 1)
        cout << "Monday\n";
    else if (day == 2)
        cout << "Tuesday\n";
            else if (day == 3)
                cout << "Wednesday\n";
     // You may complete it yourself
}
```

The if/else Selection Structure

Nested if/else structures

- In previous example, the nested if/else structure is to be used.
- Test for multiple cases by placing if/else selection structures inside if/else selection structures
- Deep indentation usually not used in practice
- How can we solve this example more conveniently?
- **Switch statement is a convenient way for it.**

switch statements

Switch Statement

If you have a large decision tree, and **all the decisions depend on the value of the same variable**, you will probably want to consider a switch statement instead of a **ladder** of if...else or else if constructions.

// Example of switch statement

```
#include <iostream>
using namespace std;
int main()
{
    int day;
    cout << "The day number 1 means Monday\n";
    cout << "Please enter the number of day. \n The
    number must be any integer value from 1 to 7\n";
    cin >> day;
    // Contd. (next page)
```

```
switch(day)
{
	case 1:
		cout << "Monday\n";
		break;
	case 2:
		cout << "Tuesday\n";
		break;
	// please write cases 3 to 6 yourself
	case 7:
		cout << "Sunday\n";
		break;
	default:
		cout << "The number must be any integer value from 1 to 7\n";
		break;
}
getch();
return 0;
}
```

The switch Structure

- **switch**
  - Useful when a variable or expression is tested for all the values which can happen and different actions are taken
- **Format**
  - Series of case labels and an optional default case

```
switch ( value )
{
        case '1':
                actions
        case '2':
                actions
        default:
                actions
}
```

  - **break**; exits from structure

The switch Structure

- Flowchart of the switch structure