Introduction to C++ and differences between C and C++

BY PRAPHUL KOLTE

Introduction

- History of C Language
  - Developed by **Mr. Dennis MacAlistair Ritchie** (American Computer Scientist)
  - Development years: 1969-1972
  - Standards- C89, C90, C99, C11 etc
- History of CPP Language
  - Developed by **Mr. Bjarne Stroustrup (**Danish computer scientist)
  - Standards- C++98, C++03, **C++11, C++14,C++17 etc**

Introduction

- C++ is derived from C Language. It is a Superset of C.
- Earlier C++ was known as C with classes.
- In C++, the major change was the addition of classes and a mechanism for inheriting class objects into other classes.
- Most C Programs can be compiled in C++ compiler.
- C++ expressions are the same as C expressions.
- All C operators are valid in C++.

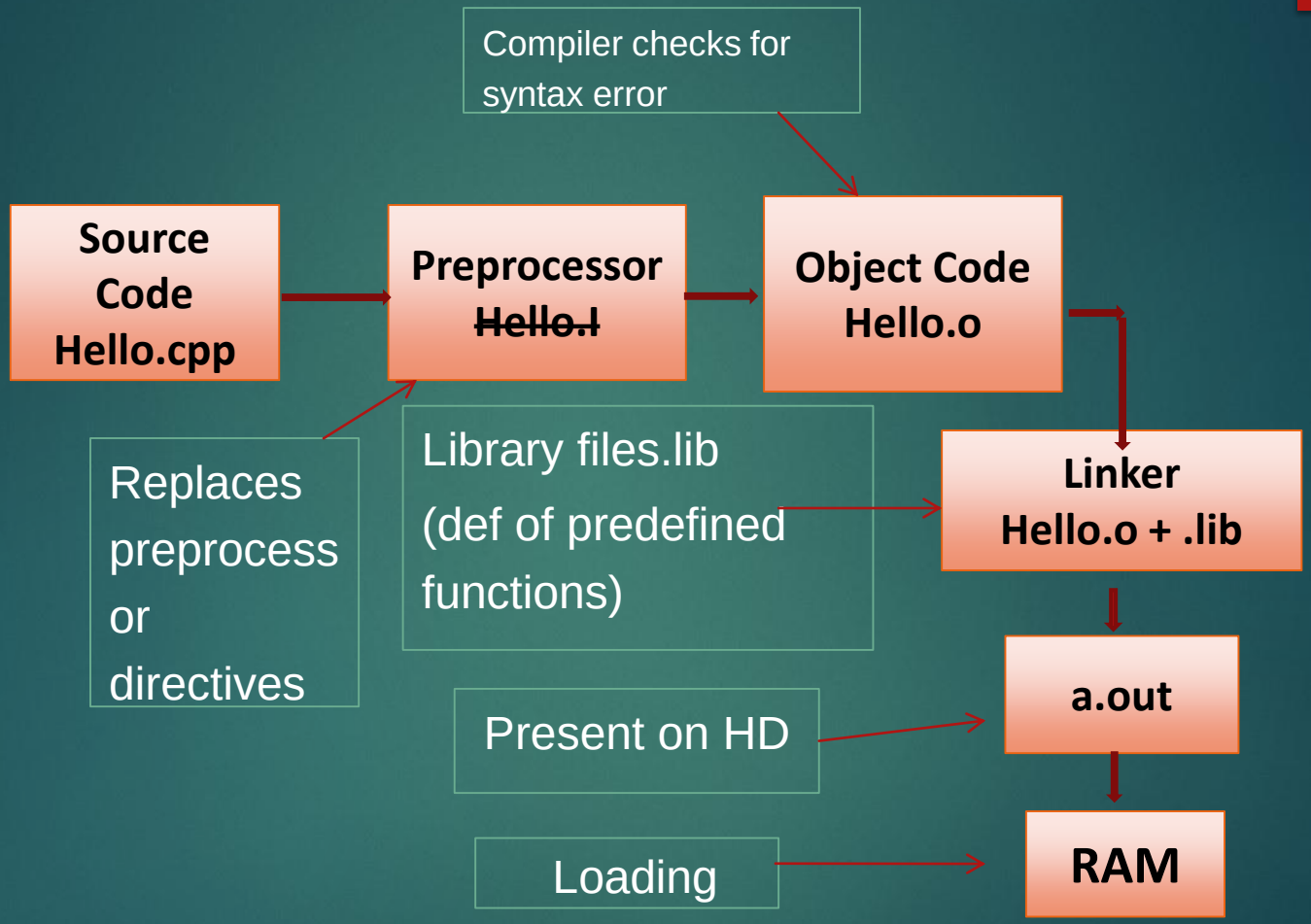
Hello World in C++


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

Compilation:
on Linux Platform
g++ Hello.cpp
After compilation it will produce a.out file
Execution: ./a.out

- cout is object of ostream
- cin is object of istream
- **>>** called as **extraction** or input operator
- **<<** is called as **insertion** operator or output operator
- Using namespace **std** is used to specify namespace
- **returning 0 tell OS that program execution is graceful**

Compilation process



Preprocessor

- Preprocessor executes before compilation
- Preprocessor directive starts with #
- It is mainly used for
  - **File inclusion**

    #include “filename”, #include<filename>
  - **Macro substitution**

    #define PI 3.14
  - **Conditional compilation**

```
#define DEBUG

#ifdef DEBUG
  //Any code
#endif
```

```
#ifndef DEBUG
  //Any code
#endif
```

Compiler

- Compiler translates source code from a high-level programming language to a lower level language (e.g., assembly language, object code, or machine code) to create an executable program
- Compiler does following tasks
  - **Scanning**:
    - Reads one character at a time from the source code
    - Keeps track of which character is present in which line.
  - **Lexical Analysis**:
    - The compiler converts the sequence of characters that appear in the source code into a series of strings of characters (known as tokens)
    - Tokens are associated by a specific rule by a program called a Lexical Analyzer.
    - A symbol table is used by the Lexical Analyzer to store the words in the source code that correspond to the token generated.

Compiler

- **Syntactic Analysis:**
  - Syntax analysis involves preprocessing to determine whether the tokens created during lexical analysis are in proper order as per their usage.
  - The correct order of a set of keywords, which can yield a desired result, is called syntax. The compiler has to check the source code to ensure syntactic accuracy.
- **Semantic Analysis:**
  - The structure of tokens is checked, along with their order with respect to the grammar in a given language.
  - The meaning of the token structure is interpreted by the parser and analyser to finally generate an intermediate code, called object code.
  - Finally, the entire code is parsed and interpreted to check if any optimizations are possible. If Possible optimization will be done

GCC: GNU Compiler Collection

- GCC used for C, C++, Java etc.
- GCC was first developed for GNU Operating System
- Latest available version of GCC is 9.2

Examples:

Language	Sample Command
C	**gcc** Hello.c
Cpp	**g++** Hello.cpp
Java	**gcj** Hello.java

- Up to gcc6.1 Default cpp version in **C++98**
- gcc6.1 has C++14 as default version
- In gcc4.1 support for C++11 added

Linker

Linker is a computer utility program that takes one or more object files generated by a compiler and combines them into a single executable file, library file, or another 'object' file.

Types of Linking

- Static Linking (compile time)

  Ex *.lib

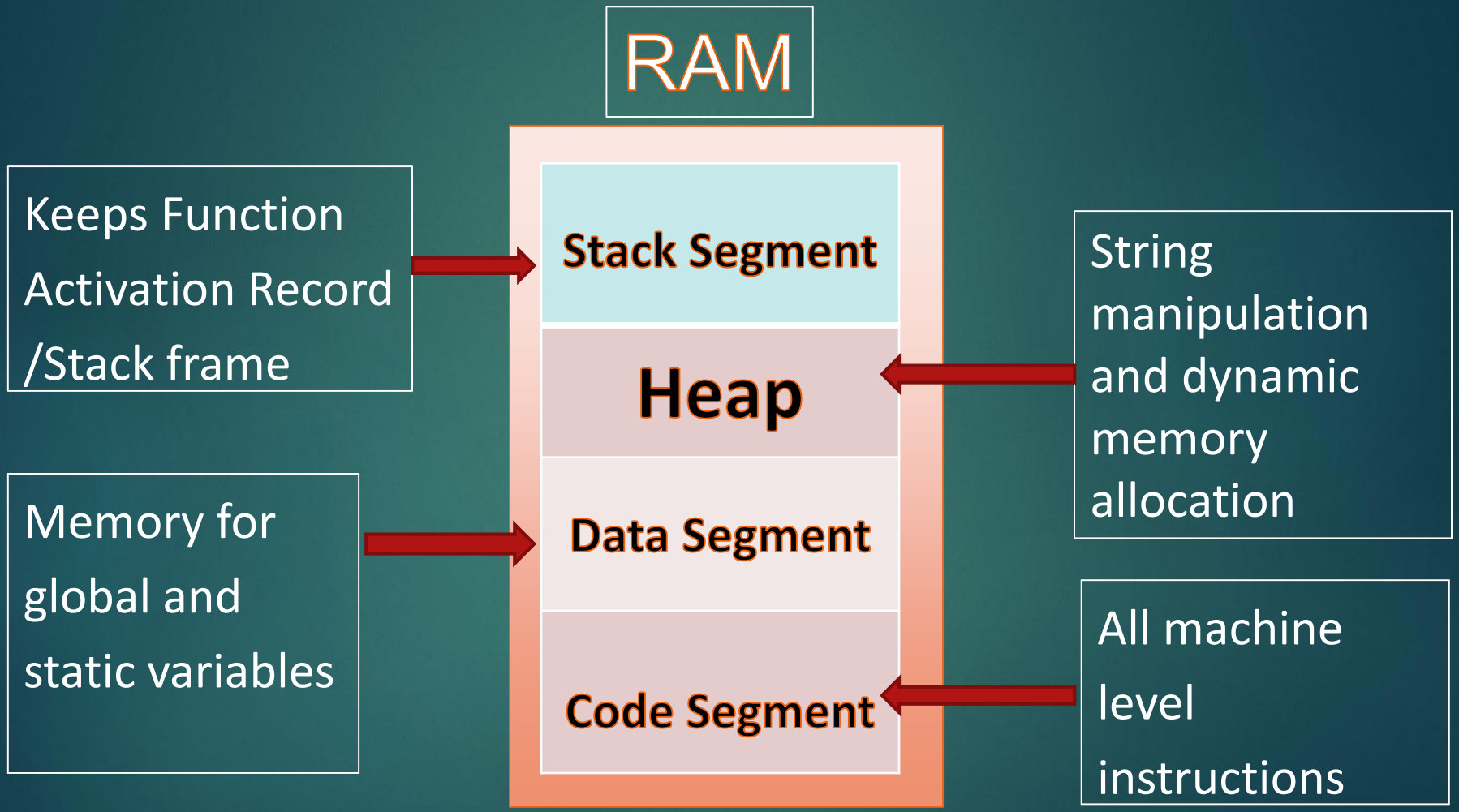
- Dynamic Linking (run time)

  Ex. *.dll

Program memory

- When we run a program ,os allocates part of memory for that program and then copies executable from disk to memory.
- The C compiler divides this area in 4 parts
  - Stack
  - Heap
  - Code Area/Segment
  - Data Area/Segment
    - Initialized data
    - Uninitialized data

Program Memory



Code Segment

- Fixed in nature because size of program is known at load time.
- **It is reserved for executable code of program**.
- This is read only memory area we can not change it during execution.
- Only Pointers to functions can access this area.

Data Segment

- Fixed in nature because size of programs data is known at load time.
- It contains internal and external static variables, global variables, initialized array and structures and constant strings.
- **Initialized Data Area:**
  - **On Initialization static and global variables are stored at initialized data area.** All other variables gets stored in uninitialized data area.
- **Uninitialized Data Area:**
  - In uninitialized data area variables get initialized to 0 nut in initialized area they are initialized with their respective values.
  - **Static and global variables known as load time variables.**

Keywords from C to C++

auto const double float int
short struct unsigned break continue
else for long signed switch void
case default enum goto register
sizeof typedef volatile char do
extern if return static union
while

Keywords in C++

30 reserved words that were not in C:

asm public private protected new delete bool false true try catch throw class this friend template using namespace inline typename typeid wchar_t explicit virtual mutable operator static_cast dynamic_cast reinterpret_cast const_cast

Predefined identifiers

cin endl INT_MIN iomanip main
npos std **cout** include INT_MAX
iostream MAX_RAND NULL string

Difference between C and C++

C11 Programming Language	C++11 Programming Language

C is Procedural Language	C++ is Object Oriented Language.
	It supports Object Oriented principle like Encapsulation, Inheritance, Polymorphism, Coupling and Cohesion.
	
	C++ is treated as not pure Object Oriented Lang.
	Reason: We can write C++ program without writing class and creating Object and Friend functions
	
	Example:
	`#include<iostream>`
	`using namespace std;`
	`int main()`
	`{`
	`cout<<"\nHello World\n";`
	`return 0;`
	`}`

Difference between C and C++

C11 Programming Language	C++11 Programming Language
Use of function before its declaration generates **WARNING (Program will run)** ***#include<stdio.h>*** ***void func1(){*** ***printf("\n func1 called");*** ***func2(); }*** ***void func2(){*** ***printf("\n func2 called"); }*** ***int main() {*** ***printf("\n main called");*** ***func1();*** ***return 0; }***	Use of function before its declaration generates **ERROR** ***#include<iostream>*** ***using namespace std;*** ***void func1(){*** ***cout<<"\n func1 called";*** ***func2(); }*** ***void func2() {*** ***cout<<"\n func2 called"; }*** ***int main(){*** ***cout<<"\n main called";*** ***func1();*** ***return 0; }***

Difference between C and C++

C11 Programming Language	C++11 Programming Language
In C, malloc() and calloc() Functions are used for Memory Allocation and free() function for memory Deallocating.	In C++, new and delete operators are used for Memory Allocating and Deallocating
In C, malloc,calloc, realloac are functions and we need to include stdlib.h header to use them.	In C++, new and delete are part of language as these are operators in C++, we don't need any header to be include to use them.
In C, malloc,calloc and realloc does not understand datatypes. These functions takes size as input. Example: *int *p= **malloc(sizeof(int));*** Note: we are passing no of bytes to be allocated to malloc	In C++, new operator understand datatype, it allocates memory and calls contrsuctor for datatype . Size is not required to be supplied Example: *int * p = **new int;*** Note: We are just specifying what is datatype.

Difference between C and C++

C11 Programming Language	C++11 Programming Language
In C, no special memory deallocation of array Example: **For single int** *int *p = malloc(sizeof(int));* ***free(p);*** Example: **For array of 10 int** *int *p = malloc(10*sizeof(int));* ***free(p);***	In C, special memory deallocation of array Example: **For single int** *int *p = new int;* ***delete p;*** Example: **For array of 10 int** *int *p = new int[10];* ***delete []p;***
Top down approach is used in Program Design. i.e. Program to functions	Bottom up approach adopted in Program Design i.e. Objects to functionality

Difference between C and C++

C11 Programming Language	C++11 Programming Language
In C, there is no mechanism for scope resolution ***int i=10;*** ***int main()*** ***{int i=20;*** ***i = i+i;*** ***printf("\n%d",i);*** ***return 0;*** ***} // print 40***	In C++, there is scope resolution operator which help in specifying scope ***#include<iostream>*** ***using namespace std;*** ***int i=10;*** ***int main()*** ***{int i=20;*** ***i = i+ ::i;*** ***cout<<i;*** ***return 0;*** ***} // print 30***

Difference between C and C++

C11 Programming Language	C++ 11 Programming Language

C constants are compile time constants they can be changed at run time.
Example 1: (Compile time modification check)
const int k=10;
k= k+1; //Compilation error
Example 2: (Run time modification check)
#include<stdio.h>
int main(){
const int k=10;
int* p =(int*)&k;
****p=100;***
printf("\n%d",*p); // will print 100
printf("\n%d",k); // will print 100
return 0;}
	C++ constants are true constants they cannot be changed at run time.
	Example 1: (Complie time modification check)
	const int k=10;
	k= k+1; //Compilation error
	Example 2: (Run time modification check)
	#include<iostream>
	using namespace std;
	int main(){
	const int k=10;
	int* p =(int*)&k;
	****p=100;***
	cout<<*p<<endl; // will print 100
	cout<<k<<endl; // will print 10
	return 0;}

Difference between C and C++

C11 Programming Language	C++11 Programming Language
C uses pointers for memory handling	C++ minimizes use of pointers by introducing references

In C, function call cannot be written on LHS of assignment operator.	In C++, function call can be written on LHS of assignment operator.
	`#include<iostream>`
	`using namespace std;`
	`int k = 10;`
	`int& function()`
	`{ return k;}`
	`int main(){`
	`function()=100;`
	`cout<<k; // Will print 100`
	`return 0;}`

Difference between C and C++

C11 Programming Language	C++11 Programming Language

Placeholder arguments in functions are not allowed.	Function can have placeholder arguments
	Example:
	int function(int a, int b, int)
	{
	return a+b;
	}

Namespaces are not allowed to separate scope.	Namespaces can be used to separate out scope.
String handling is done using char array	String handling can be done using char array but C++ provides simple to use string class to handle strings.
C struct does not support Encapsulation	C++ struct supports Encapsulation

The C++ string class

- Must #include <string> to create and use string objects
- Can define string variables in programs

```
string name;
```

- Can assign values to string variables with the assignment operator

```
name = “Alia";
```

- Can display them with cout

```
cout << name;
```

- Can input string with cin

```
cin >>name;
```

Thank You

Feel the difference!!!! ……C++