

# Object Oriented Programming Using C++

Ketan G Kore

[ketan.kore@sunbeaminfo.com](mailto:ketan.kore@sunbeaminfo.com)

# Programming Language

- Business logic implementation
- Application Development( CUI, GUI, Library application etc. )
- Building blocks of the language
  1. Syntax and Semantics
  2. Data types
  3. Tokens
  4. Built-in features
  5. Standard library and run-time system
- Examples
  - C, C++, Java, C#, Python, Go, Scala etc.

# Token

- Program is collection of statements.
- Statement is an instruction given to the computer.
- Every instruction is made from token.
- Token is basic unit of a program.
- Following are tokens in C:
  1. Identifier
  2. Keyword
  3. Constant / Literal
  4. Operator
  5. Punctuator/Separator

# Types of programming languages

1. Machine level programming language.
2. Low level programming language.
3. High level programming language.

# Programming Language Paradigms

## 1. Procedure Oriented Programming Languages.

- FORTRAN, ALGOL, COBOL, BASIC, Pascal, C etc.

## 2. Object Oriented Programming Languages.

- **Simula**, Smalltalk, C++, Java, C# etc.

## 3. Object Based Programming Languages.

- **Ada**, Modula-2, Visual Basic, Java Script

## 4. Functional Oriented Programming Languages.

- **LISP**, Python, Scala, Haskell etc.

# Object Oriented Programming Structure

- “Object-Oriented Programming” (OOP) was coined by **Alan Kay**.
- According to Alan Kay, the essential ingredients of OOP are:
  - Message passing
  - Encapsulation
  - Dynamic binding
- Grady Booch is inventor of UML( Unified Modelling Language ).
- According to Grady Booch, there are 4 major and 3 minor pillars of oops.

# Major Pillars Of OOPS

- Following are the four major pillars of oops:
  1. Abstraction – To achieve simplicity
  2. Encapsulation – To achieve data hiding
  3. Modularity – To minimize module dependency
  4. Hierarchy – To achieve reusability
- By major, we mean that a language without any one of these elements is not object oriented.

# Minor Pillars Of OOPS

- Following are the three **minor pillars** of oops:
  1. **Typing** – To reduce maintenance of the system
  2. **Concurrency** – To utilize hardware resources efficiently
  3. **Persistence** – To maintain state of object on secondary storage.
- By minor, we mean that each of these elements is a useful, but not essential, to classify language object oriented.

# C++ Introduction

- C++ is a general-purpose programming language created by Bjarne Stroustrup in 1979.
- It is a pure C programming language in addition with Classes.
- It is object oriented programming language which is derived from C and Simula.
- Extension of C++ source file should be .cpp.
- Initial name of the language was “C with Classes”.
- C++ is standardized by the International Organization for Standardization(ISO).
- C++98, C++03, C++11, C++14, C++17, C++20, C++23 are C++ standards.
- Reference Website : <https://en.cppreference.com/w/cpp>

# C++ Programming: Language Keywords

- Keywords are the reserved words that we can not use as identifier
- Reference : <https://en.cppreference.com/w/cpp/keyword>
- According to C++ 98, there are 74 keywords in C++.
  - C++98 : 74 keywords
  - C++11 : 10 keywords

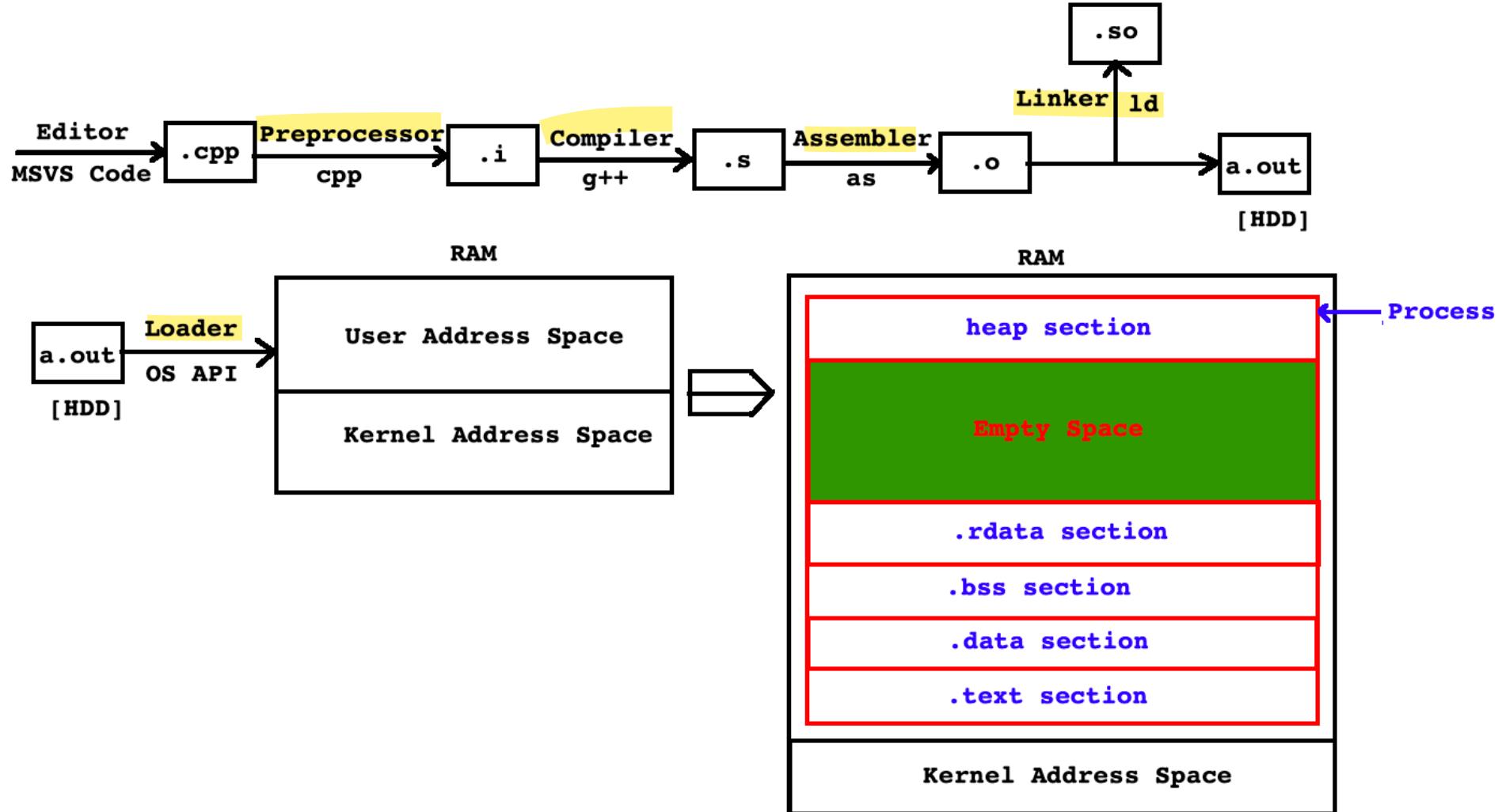
# Data Type

- Data type of any variable decides 4 things:
  1. Memory
  2. Nature
  3. Operation
  4. Range
- Types of data type:
  1. Fundamental data types
  2. Derived data types
  3. User Defined data types

# Data Type

Fundamental Data Types	Derived Data Types	User Defined Data Types
void(Not Mentioned)	Array	enum
bool( 1 byte )	Function	union
char( 1 byte)	Pointer	structure
wchar_t( 2 bytes )	Reference	class
int( 4 bytes )		
float( 4 bytes )		
double( 8 bytes )		

# Flow of execution



# Flow of execution

- “.text” section : It contains the executable instruction codes and is shared among every process running the same binary.
- “.bss” section : BSS stands for ‘Block Started by Symbol’. It holds un-initialized global and static variables.
- “.data” section : Contains the initialized global and static variables and their values. It is usually the largest part of the executable. It usually has READ/WRITE permissions.
- “.rdata” section : Also known as .rodata (read-only data) section. This contains constants and string literals.
- “Heap section” : It is used to allocate memory for variables dynamically.
- “Empty space” : Since size of stack and heap is not fixed, it may grow/shrink at runtime.

# Access Specifier

- If we want to control visibility of the members of a structure/class then we should use access specifier.
- Private, protected and public are access specifiers in C++.

<b>Access Specifier</b>	<b>Same Class</b>	<b>Derived Class</b>	<b>Non Member Function</b>
<b>private</b>	<b>A</b>	<b>NA</b>	<b>NA</b>
<b>protected</b>	<b>A</b>	<b>A</b>	<b>NA</b>
<b>public</b>	<b>A</b>	<b>A</b>	<b>A</b>

- In C++, structure members are by default public and class members are by default private.

# Data Member

- A variable declared inside class/class scope is called data member.
- Data member is also called as field/attribute/property.

```
class Complex{  
private:  
    int real; //Data Member  
    int imag; //Data Member  
};
```

- Only data member get space once per object and according to order of their declaration inside class.
- Only by understanding problem statement, we can decide data members inside class/structure.

# Member Function

- A function defined/implemented inside class/class scope is called member function.
- Member function is also called as method/operation/behavior/operation.

```
class Complex{  
public:  
    void acceptRecord( void ){ //Member Function  
    }  
    void printRecord( void ){ //Member Function  
    }  
};
```

- Member function do not get space inside object. All the objects of same class share single copy of member function.

# Class

- class is a keyword in C++.
- It is a collection of data member and member function.
- It is a basic unit of encapsulation.
- Class can contain:
  - Data members
    - 1. Non static data members
    - 2. Static data members
  - Member Functions
    - 1. Non static member functions
    - 2. Static member functions
  - Nested Types

# Class

- Some member functions are special: under certain circumstances they are defined by the compiler:
  1. **Constructor**
  2. **Destructor**
  3. **Copy constructor**
  4. **Assignment operator function**
- A class from which, we can create object is called concrete class. In words, we can instantiate concrete class.
- A class from which, we can not create object is called abstract class. In words, we can not instantiate abstract class.
- A member function of a class, which is having a body is called concrete method.
- A member function of a class, which does not have a body is called abstract method.

# Instance and Instantiation

- Variable or instance of a class is called object.
- Process of creating object from a class is called instantiation.
- Syntax :
  1. class ClassName identifier; //or
  2. ClassName identifier;
- As shown above, during instantiation, use of class keyword is optional.
- Example:
  - Complex c1;
  - Here class Complex is instantiated and name of instance is c1.
- Global variable/Local Variable/Function Parameter / Member function do not get space inside object.

# Message Passing

- Process of calling member function on object is called message passing.
- Consider following example:

```
Complex c1;  
c1.acceptRecord( ); //Message Passing  
c1.printRecord( ); //Message Passing
```

- or

```
Complex c1;  
c1.Complex::acceptRecord( ); //Message Passing  
c1.Complex::printRecord( ); //Message Passing
```

**Thank you**