Agenda

- Language Fundamentals
- Introduction to CPP
- OOP Concepts
- Hello World
- Data types
- Structure in C++

Programming Language

- A langugage that is used to communicate with the computers is called as programming language
- Computer is a hardware which understands only binary form i.e 0 or 1
- Generally these are nothing but some volts of current which gives out necessary instruction to the computer
- It is not possible for us to convert every instructions to binary and input it to CPU, and hence to make developer life simpler the programming languages are created.
- Assembly is one of the low level languguage that workes on the basics of opcode to provide the instruction to the CPU
- The langugages are classified as

1. Low level languages
   1. Machine level languages Binary language(1, 0)
   2. Assembly language
2. High level languages
   1. compiled (C, CPP)
   2. interpreted (HTML,JS)

Classfication of high languages:

1. Procedure Oriented Programming Languages
   - PASCAL, FORTRAN, COBOL, C, ALGOL, BASIC etc.
   - FOTRAN is first high level pop language.
2. Object Orineted Programming Languages
   - Simula, Smalltalk, C++, Java, Python, C# etc.
   - Simula is first object oriented programming language. It is developed in 1960 by Alan kay.
   - Smalltalk is first pure object oriented programming language which is developed in 1967.
   - More 2000 languages are object oriented.
3. Object based programming languages
   - Ada, Modula-2, Java Script, Visual Basic etc.
   - Ada is first object based programming language.
4. Functional programming languages Java, Python etc.

Chracteristics of Language

1. It has own syntax
2. It has its own rule(semantics)

3. It contain tokens:
    1. Identifier
    2. Keyword
    3. Constant/literal
    4. Operator
    5. Seperator / punctuators
4. It contains built in features.
5. We use language to develop application(CUI, GUI, Library)
6. If we want to implement business logic then we should use language.

ANSI

- Set of rules is called standard and standard is also called as specification.
- American National Standard Institute(ANSI) is an organization which is responsible for standardization of C/C++ and SQL.
- ANSI is responsible for updating language ie. adding new features, updating existing features, deleting unused features.

Histroy of C++

- Inventor of C++ is Bjarne Stroustrup.
- C++ is derived from C and simula.
- Its initial name was "C With Classes".
- It was developed in "AT&T Bell Lab" in 1979.
- It was developed on Unix Operating System.
- Standardizing C++ is a job of ANSI.
- In 1983 ANSI renamed "C With Classes" to C++.
- C++ is objet orieted programming language.
- In C++ we can develop code using Procedure as well as object orieneted fashion. Hence it is also called Hybrid programming language.

C++ Standards

- In 1985, the first edition of The C++ Programming Language was released.
- In 1989, C++ 2.0 was released. New features in 2.0 included multiple inheritance, abstract classes, static member functions, const member functions, and protected members.Later feature additions included templates, exceptions, namespaces, new casts, and a Boolean type.
- In 1998, C++98 was released, standardizing the language, and a minor update (C++03) was released in 2003.
- After C++98, C++ evolved relatively slowly until, in 2011, the C++11 standard was released, adding numerous new features,enlarging the standard library further, and providing more facilities to C++ programmers.
- A minor C++14 update was released in December 2014.
- A major revision where various new additions were introduced in C++17.
- Year and version of cpp
    1. 1998 : C++98
    2. 2003 : C++03
    3. 2011 : C++11

4. 2014 : C++14
5. 2017 : C++17
6. 2020 : C++20

Object-oriented software development (OOSD)

- In the past, the problems faced by software development were relatively simple, from task analysis to programming, and then to the debugging of the program, if its not too big it can be done by one person or a group.
- With the rapid increase of software scale, software personnel faces the problem that is very complicated, and there are many factors that need to be considered.
- The errors generated and hidden errors may reach an astonishing degree, this is not something that can be solved in the programming stage.
- Need to standardize the entire software development process and clarify the software
- The tasks of each stage in the development process, while ensuring the correctness of the work of the previous stage, proceed to the next stage work.
- This is the problem that software engineering needs to study and solve.
- Object-oriented software development and engineering include the following parts:

1. Object oriented analysis (OOA)

- The first step of Object-oriented software development is Object-Oriented Analysis (OOA)
- In the system analysis stage of software engineering, system analysts must integrate with users to make precise Accurate analysis and clear description, summarize what the system should do (not how) from a macro perspective.
- Face right the analysis of the image should be based on object-oriented concepts and methods.
- In the analysis of the task, from the objective existence of things and the relationship between the related objects (including the attributes and behaviors of the objects) and the relationship between the objects are summarized.
- The Objects with the same attributes and behaviors are represented by a class.
- Establish a need to reflect the real work situation model. The model formed at this stage is relatively rough (rather than fine).

2. Object oriented design (OOD)

- The second step of Object-oriented software development is Object-Oriented Design (OOD).
- According to the demand model formed in the object-oriented analysis stage, each part is specifically designed.
- The design of the line class may contain multiple levels (using inheritance).
- Then these classes put forward the ideas and methods of program design, including the design of algorithms.
- In the design stage no specific plan is involved, but a more general description tool (such as pseudo code or flowchart)is used to describe.

3. Object-oriented programming (OOP)

- The third step of Object-oriented software development is Object-oriented Programming (OOP).

- According to the results of object-oriented design, to write it into a program in a computer language, it is obvious that object-oriented Computer language (e.g. C++) needs to be used.
- Otherwise the requirements of object-oriented design cannot be achieved.

Object-oriented programming

- OOPS is not a syntax. It is a process / programming methodology which is used to solve real world problems.
- It is invented by Dr. Alan Kay. He is inventor of Simula too.
- Unified Modelling Language(UML) is invented by Grady Booch. If we want to do OOA and OOD then we can use UML.
- According Grady Booch there are 4 main/major and 3 minor elements/parts/pillars of OOPS
- 4 major pillars of oops

1. Abstraction
2. Encapsulation
3. Modularity
4. Hierarchy

- 3 minor pillars of oops

1. Typing
2. Concurrency
3. Persistence

- Here, word major means, language without any one of the above feature will not be Object orieneted.
- Here word minor means, above features are useful but not essential to classify language object oriented.

Abstraction

- Getting only essential things and hiding unnecessary details is called as abstraction.
- Abstraction always describe outer behavior of object.
- In console application when we give call to function in to the main function , it represents the abstraction

Encapsulation

- Binding of data and code together is called as encapsulation.
- Implementation of abstraction is called encapsulation.
- Encapsulation always describe inner behavior of object
- Function call is abstraction
- Function definition is encapsulation.

Modularity

- Dividing programs into small modules for the purpose of simplicity is called modularity.

Hirerachy

- Level / order / ranking of abstraction is called hireracy.
- Its main purpose is to achive reusability.
- Advantages of reusability:

1. To reduce develoers efforts.
2. To reduce development time and development cost.

- Types of Hierarcy:

1. Has-a/Part-of Association/Containment
2. Is-a/Kind-of Inheritance/Generalization
3. Use-a Dependancy
4. Creates-a Instantiation

Typing/Polyorphism

- polymorphism = poly(many) + morphism(forms)
- An ability of object to take multiple forms is called polymorphism.
- Main purpose of polymorphism is to reduce maintenance of system.
- Types of polymoprhism:

1. Compile time polymoprhism
   - It is also called as static polymoprhism / Early Binding / False polymoprhism / Weak Typing
   - We can achive it using:
     1. Function Overloading
     2. Operator Overloading
     3. Template
2. Runtime polymorphism
   - It is also called as dynamic polymoprhism / Late Binding / True polymoprhism / Strong Typing
   - We can achive it using:
   1. Function Overriding

Concurrency

- In context of OS it is called multitasking
- Process of executing multiple task simultaneously is called concurrency.
- If we want to utilize hardware resources efficiently then we should use concurrency.
- Using multithreading, we can achive concurrency.

Persistance

- It is the process of maintaing state of object on secondry storage(HDD).
- We can achive it using file handling and database programming.

Installation on linux

- To install the gcc/g++ compiler use the below commands

```
sudo apt update
sudo apt install build-essential
gcc --version
g++ --version
```

- Download the vsCode form the ubuntu snap store

Installation on windows

- Download the gcc/g++ compiler use the below link

```
https://sourceforge.net/projects/gcc-win64/
```

- Download the vs code from microsoft store or you can also use the below link

```
https://code.visualstudio.com/download
```

Hello World

```
// header file
#include <iostream>

// Entry point function
int main()
{
    printf("Hello World");
    return 0;
}
```

- Steps for the compilation

```
g++ demo01.cpp
./a.out
```

Execution of a C++ program

- It involves four stages using different compiling/execution tool, these tools are set of programs which help to complete the C/C++ program's execution process.

1. Preprocessor
    - This is the first stage of any C/C++ program execution process; in this stage Preprocessor processes the program before compilation. Preprocessor include header files, expand the

Macros.

2. Compiler
    - This is the second stage of any C/C++ program execution process, in this stage generated output file after preprocessing (with source code) will be passed to the compiler for compilation. Complier will compile the program, checks the errors and generates the object file (this object file contains assembly code).
3. Linker
    - It will link the multiple files
4. Loader
    - It will load the executable for the execution

- We can view the intermediate files like preprocesssed (.i),assembly (.asm) file for out .cpp using below commands
- we can view the .i and .asm files but we cannot view the object .o and executable .exe files

```
# creates an preprocessed File
g++ -E demo01.cpp -o demo.i

# creates an Assembly File
g++ -S demo01.cpp -o demo.asm

# cretaes an object file
g++ -c demo01.cpp

# cretaes an executable file
g++ demo01.cpp
```

Data Types in cpp

- It describes 3 things about variable / object

    1. Memory : How much memory is required to store the data.
    2. Nature : Which type of data memory can store
    3. Operation : Which operations are allowed to perform on data stored inside memory.

- Their are 3 types of Data types in cpp

1. Fundamental Data types(7)

    1. void : Not Specified
    2. bool : 1 byte
    3. char : 1 byte[ASCII]
    4. wchar_t : 2 bytes[Unicode]
    5. int : 4 bytes
    6. float : 4 bytes
    7. double : 8 bytes

2. Derived Data types(4)

    1. Array
    2. Function
    3. Pointer
    4. Reference

3. User Defined Data Types(3)

    1. Union
    2. Structure
    3. Class

Type Modifiers

- C++ allows the char, int, and double data types to have modifiers preceding them.
- A modifier is used to alter the meaning of the base type so that it more precisely fits the needs of various situations.
- The data type modifiers are
    1. short
    2. long
    3. signed
    4. unsigned
- The modifiers signed, unsigned, long, and short can be applied to integer base types. In addition, signed and unsigned can be applied to char, and long can be applied to double.
- The modifiers signed and unsigned can also be used as prefix to long or short modifiers. For example, unsigned long int.

Type Qualifiers

- The type qualifiers provide additional information about the variables they precede.
- their are two qualifiers

1. const
2. volatile

Bool Datatype

- it can take true or false value.
- It takes one byte in memory.
- by default the value is false

wchar_t Datatype

- wchar_t stands for Wide Character.
- This should be avoided because its size is implementation defined and not reliable.
- It is similar to char data type, except that it take up twice the space and can take on much larger values.
- As char can take 256 values which corresponds to entries in the ASCII table. On the other hand, wide char can take on 65536 values which corresponds to UNICODE values which is a recent international standard which allows for the encoding of characters for virtually all languages and commonly used symbols.

- The type for character constants is char, the type for wide character is wchar_t. This data type occupies 2 or 4 bytes depending on the compiler being used. Mostly the wchar_t datatype is used when international languages like Japanese are used. This data type occupies 2 or 4 bytes depending on the compiler being used. L is the prefix for wide character literals and wide-character string literals which tells the compiler that that the char or string is of type wide-char.

Structure in CPP

- In Cpp we can define the functions within the structure
- to access the members of structure we have to create the variable of the structure and access the members using .operator.

```
struct Time
{

    int hrs;
    int min;

    void acceptTime()
    {
        printf("Enter hrs and mins - ");
        scanf("%d%d", &hrs, &min);
    }

    void printTime()
    {
        printf("Time - %d : %d \n", hrs, min);
    }
};

int main()
{
    struct Time t1;
    t1.acceptTime();
    t1.printTime();
    return 0;
}
```