**Sunbeam Infotech**

**CPP Notes - Day 1**

**\* Few Real Time Applications**

Games

GUI Based Application (Adobe)

Database Software (MySQL Server)

OS (Apple OS)

Browser( Mozilla)

Google Applications(Google File System and Chrome browser)

Banking Applications

Compilers

Embded Systems(smart watches, MP3 players, GPS systems)

**\* 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 )

**\*SDK (Software Development Kit)**

- SDK = Language tools + Documentation + Supporting Library + Runtime Env.

# Language Tools:

-Editor (to develop/edit source code)

-Preprocessor (To remove preprocessors-remove comments/expand macros)

-Compiler (Conversion of high level language into low level code( Assembly ))

-Assembler (Conversion of low level code into machine code)

-Linker

-Loader

-Debugger

# Documentation (MSDN / man pages)

**\* History of C++**

-Inventor of C++ is Bjarne Stroustrup.

-C++ is derived from C and simula.

-Its initial name was "C With Classes".

-At is developed in "AT&T Bell Lab" in 1979.

-It is developed on Unix Operating System.

-In 1983 ANSI renamed "C With Classes" to C++.

-C++ is objet orieted programming language

**\*Object Oriented Programming Structure**

-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.**

**\* Data Type**

- 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.

**\* Data Types in C++**

- Fundamental Data Types (void, int,char,float,double)

-Derived Data Types ( Array, Function, Pointer, Union Structure)

**\* Variable Declaration and Definition**

-Variable Initialization and variable assignment

**\* Main Function**

- main should be entry point function of C/C++

-Calling/invoking main function is responsibility of operating system. Hence it is also called as Callback function

**\* Structure in C & C++**

|  |  |
| --- | --- |
| struct in c | struct in c ++ |
| we can include only variables into the structure. | we can include the variables as well as the functions in structure. |
| Writing struct keyword is compulsory | Writing struct keyword is optional |
| We need to pass a structure variable by value or by address to the functions. | We don't pass the structure variable to the functions to accept it / display it.The functions inside the struct are called with the variable and DOT operator. |
| By default all the variables of structure are accessible outside the structure. | By default all the members are accessible outside the structure, but we can restrict their access by applying the keywords private /public/ protected. |
| struct Time t1; | struct Time t1; |
| AcceptTime(struct Time &t1); | t1.AcceptTime(); //function call |

**\* Function Declaration ,Defination and call**

**\* Inline Function**

- inline function is commonly used with classes.

- If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.

\*When to use Inline function?

- We can use Inline function as per our needs.

- We can use the inline function when performance is needed.

- We can use the inline function over macros.

- We prefer to use the inline keyword outside the class with the function definition to

hide implementation details of the function.

**\* Default Arguments**

- A default argument is a default value provided for a function parameter.

- If the user does not supply an explicit argument for a parameter with a default argument, the default value will be used.

- If the user does supply an argument for the parameter, the user-supplied argument is used.

**\* Function Overloading**

- same function name and diff signature of function

- Name mangling process , mangled name

- Differ in number of input arguments

- Differ in datatype of input arguments

- Differ atleast in the sequence of the input arguments

**\* Namespace**

- to prevent name conflicts/ collision / ambiguity in large projects

- to group/orgaize functionally equivalent / related types toghter.

-If we want to access value of global variable then we should use scope resolution operator ( ::)

-We can not instantiate namespace.

-It is designed to avoid name ambiguity and grouping related types.

-If we want to define namespace then we should use **namespace** keyword.

-we can not define namespace inside function/class.

-If name of the namespaces are same then name of members must be different.

We can not define main function inside namespace.

Namespace can contain:

1. Variable

2. Function

3. Types[ structure/union/class]

4. Enum

5. Nested Namespace

**\* Class and Object**

**Class:**

-Class is collection of data member and member function.

-Class represents set/group of such objects which is having common structure and common

behavior.

-class is logical entity.

- Class contain (Nested Type[enum,class,structure,union],Data Members,Member Function]

**Object :**

- Instance of class is object

-An entity, which get space inside memory is called object.

-Object is used to access data members and member function of the class

-Process of creating object from a class is called instantiation

**- Object has**

1. State

- Value stored inside object is called state of the object.

- Value of data member represent state of the object.

2. Behavior

- Set of operation that we perform on object is called behavior of an

object.

- Member function of class represent behavior of the object.

3. Identity

- Value of any data member, which is used to identify object uniquly

is called its identity.

- If state of object is same then its address can be considered as its

identity.

-Member function do not get space inside object.

-If we create object of the class then only data members get space inside object. Hence size of object is depends on size of all the data members declared inside class.

-Data members get space once per object according to the order of data member declaration.

-Structure of the object is depends on data members declared inside class.

-Member function do not get space per object rather it gets space on code segment and all the objects of same class share single copy of it.

-Member function's of the class defines behavior of the object.

**\* Data Members**

-Data member is also called as field, attribute, property etc.

**\*Member Functions**

-A function implemented inside class scope is called member function

-Member function is also called as method, operation, behavior or message.

**\*this Pointer**

-To process state of the object we should call member function on object. Hence we must define member function inside class.

-If we call member function on object then compiler implicitly pass address of that object as a argument to the function implicitly.

-To store address of object compiler implitly declare one pointer as a parameter inside member function. Such parameter is called this pointer.

-this is a keyword. "this" pointer is a constant pointer.

-this is used to store address of current object or calling object.

-Following functions do not get this pointer:

1. Global Function

2. Static Member function

3. Friend Function.

**\* Difference between <filename.h> and “filename.h”**

-"/usr/include" directory is called standard directory for header files.

-<filename.h> preprocessor try to locate and load header file from standard directory only(/usr/include).

-”filename.h” - preprocessor try to locate and load header file first from current project directory if not found then it try to locate and load from standard directory.

**\* Header Guards**

-If we want to expand contents of header file only once then we should use header guard

#ifndef HEADER\_FILE\_NAME\_H\_

#define HEADER\_FILE\_NAME\_H\_

//TODO : Type declaration here

#endif

**\*Four major pillars(language without any one of the above feature will not be Object**

**orieneted.)**

- Abstraction

- Encapsulation

- Modularity

- Hierarchy

**\*Three Minor Pillars (these features are useful but not essential to classify language object oriented)**

- Typing / Polymorphisim

- Concurrency

- Persistance