Shree Ganeshay Namh

Storage Class

- Storage Class are used to describe the characteristics of a variable / function

- storage_class d_type var= value;

Mutable Class

Thread_local

- Bool
- Wide character (wchar_t – 2 or 4)bytes

Data Types

- All like c but –
- Bool
- Wchar_t
- Refrence (int& a=var)

Data Type	Size (in bytes)	Range
float	4	-3.4×10^38 to 3.4×10^38
double	8	-1.7×10^308 to1.7×10^308
long double	12	-1.1×10^4932 to1.1×10^4932
wchar_t	**2 or 4**	**1 wide character**

<limits.h>

- *Syntax :-* <limits.h> *header file is defined to find the range of fundamental data-types. Unsigned modifiers have minimum value is zero. So, no macro constants are defined for the unsigned minimum value.*

Type-castinng

Literals

- In C & C++, Literals are the Constant values that are assigned to the constant variables. Literals represent fixed values that cannot be modified. Literals contain memory but they do not have references as variables. Generally, both terms, constants, and literals are used interchangeably.
- There are 4 types of literal in C:
    - Integer Literal
    - Float Literal
    - Character Literal
    - String Literal

Int Literal

- Integer literals are used to represent and store the integer values only.
- Integer literals are expressed in two type :-
  - i). Prefix :-
    - » Decimal - no prefix before decimal digits (0,1,2,3,4,5,6,7,8,9)
    - » Binary (base 2) ---0b or 0B before binary digits (0 / 1)
    - » Octal (base 8) ---0 before (0,1,2,3,4,5,6,7)
    - » Hexa-decimal (16) --- 0x before hexa-literal digits (0,1,2,3,4,5,6,7,8,9 ,A,a,B,b,C,c,D,d,E,e,F,f)
  - ii). Suffix :-
    - » Int --- No suffix
    - » Unsigned int -- U (int a=122345U).
    - » Long int – l,L (int a = 1234l)
    - » Long long int – ll or LL (int a=1LL)
    - » Short – No suffi x

practice

- 68   -valid
- 0876 -- invalid octal
- 0234 – valid octal
- 0xt45 -- invalid hexa-decimal

.45e10 – invalid float (exponent type)

1.45e-12 – valid float

Integer Promotion

i/o Function

- In C++ input and output are performed in the form of a sequence of bytes or more commonly known as **streams**.
- **Input Stream:** If the direction of flow of bytes is from the device(for example, Keyboard) to the main memory then this process is called input.
- **Output Stream:** If the direction of flow of bytes is opposite, i.e. from main memory to device(display screen) then this process is called output.

- In C++ after the header files, we often use ‘*using namespace std;*‘. The reason behind it is that all of the standard library definitions are inside the namespace std. As the library functions are not defined at global scope, so in order to use them we use *namespace std*.

- **Header files available in C++ for Input/Output operations are:**
- **iostream**: iostream stands for standard input-output stream. This header file contains definitions of objects like cin, cout, cerr, etc.
- **iomanip**: iomanip stands for input-output manipulators. The methods declared in these files are used for manipulating streams. This file contains definitions of setw, setprecision, etc.
- **fstream**: for file handling.
- **bits/stdc++:** This header file includes every standard library

Cin,cout,cerr,clog

- >> -- extraction operator
- << -- insertion operator
- Standard input Stream(cin)
- Standard Output Stream(cout)
- Un-buffered standard error stream (cerr):
- Buffered standard error stream (clog):
- diff b/w cout and cerr????

1.	**while loop**– First checks the condition, then executes the body.	
2.	**for loop**– firstly initializes, then, condition check, execute body, update.	
3.	**do-while loop** – firstly, execute the body then condition check	

Overloading

function

- Inline Function:-
  - Remember, inlining is only a request to the compiler, not a command. The compiler can ignore the request for inlining.
  - Compiler rejects if :-
    - If a function contains a loop. (*for, while and do-while*)
    - If a function contains static variables.
    - If a function is recursive.
    - If a function return type is other than void, and the return statement doesn’t exist in a function body.
    - If a function contains a switch or goto statement.

Lambda expression

- lambda expressions to allow inline functions which can be used for short snippets of code that are not going to be reused and therefore do not require a name. In their simplest form a lambda expression can be defined as follows:
- ***[capture clause] (parameters) -> return-type { definition of method }***

Pointers

- Null Pointer :- (value pointer)
- Void Pointer :- (type pointer)
    - It cann’t be derefrenced
    - Pointer arithmetic is not possible on pointers of void due to lack of concrete value and thus size.
- Wild Pointer :-
    - – Uninitialised pointer
- Dangling Pointer :-
    - A pointer which is free from memory using free()

Refrences

- When a variable is declared as a reference, it becomes an alternative name for an existing variable.
- Ref less powerfull than pointers.
- There are multiple applications for references in C++, a few of them are mentioned below:
    - Modify the passed parameters in a function
    - Avoiding a copy of large structures
    - In For Each Loop to modify all objects
    - For Each Loop to avoid the copy of objects

* vs &

- A pointer can be declared as void but a reference can never be void.
- The pointer variable has n-levels/multiple levels of indirection i.e. single-pointer, double-pointer, triple-pointer. Whereas, the reference variable has only one/single level of indirection.
- Reference variables cannot be updated.
- Reference variable is an internal pointer.
- Declaration of a Reference variable is preceded with the ‘&’ symbol (but do not read it as “address of”).
- Reference must be initilised while pointer not;

Pass by * vs &

Parameters	Pass by Pointer	Pass by Reference
Passing Arguments	We pass the address of arguments in the function call.	We pass the arguments in the function call.
Accessing Values	The value of the arguments is accessed via the dereferencing operator *	The reference name can be used to implicitly reference a value.
Reassignment	Passed parameters can be moved/reassigned to a different memory location.	Parameters can’t be moved/reassigned to another memory address.
Allowed Values	Pointers can contain a NULL value, so a passed argument may point to a NULL or even a garbage value.	References cannot contain a NULL value, so it is guaranteed to have some value.

Arrays

- By using refrences in C++ we remove array decay

String

- String_stream :-
    - » string_stream str_stream_obj (string);

String	Character Array
Strings define objects that can be represented as string streams.	The null character terminates a character array of characters.
No Array decay occurs in strings as strings are represented as objects.	The threat of array decay is present in the case of the character array
A string class provides numerous functions for manipulating strings.	Character arrays do not offer inbuilt functions to manipulate strings.
Memory is allocated dynamically.	The size of the character array has to be allocated statically.

String function

length()	s.length()
swap()	swap(a,b)
size()	s.size()
resize()	s.resize(10)
find() (return start_ind /npos)	s1.find(s2,s_index)
push_back()	s.push_back('c');
pop_back()	s.pop_back()
clear()	s.Clear()

strncmp() {-ive,0,+ive}	strncmp(s1,s2,num)
strncpy()	strncpy(s1,s2,num)
strrchr() {pointer of index}	strchr(s1,’k’)
strncat()	strcat(dest,src,len)
---	-----
replace() {pointer of s1}	s1.replace(strt,stop,src)
substr()	This function is used to create a substring from a given string.
compare()	This function is used to compare two strings and returns the result in the form of an integer.
erase()	This function is used to remove a certain part of a string.

Struct in C vs C++

C Structures	C++ Structures
Only data members are allowed, it cannot have member functions.	Can hold both: member functions and data members.
Cannot have static members.	Can have static members.
Cannot have a constructor inside a structure.	Constructor creation is allowed.
Direct Initialization of data members is not possible.	Direct Initialization of data members is possible.
Writing the 'struct' keyword is necessary to declare structure-type variables.	Writing the 'struct' keyword is not necessary to declare structure-type variables.
Do not have access modifiers.	Supports access modifiers.
Only pointers to structs are allowed.	Can have both pointers and references to the struct.

Structures

- *A Structure is not secure and cannot hide its implementation details from the end user while a class is secure and can hide its programming and designing details.*
- *C++ structures give access modifiers*
    - Public
    - Private
    - protected

Enumerators

- Enumerator is a special kind of data type defined by the user. It consists of constant integrals or integers that are given names by a user. The use of enum in C to name the integer values makes the entire program easy to learn, understand, and maintain by the same or even different programmer.

Exception Handling

File Handling

- STEP 1-Naming a file
  STEP 2-Opening a file
  STEP 3-Writing data into the file
  STEP 4-Reading data from the file
  STEP 5-Closing a file.
- Streams :- Streams are nothing but it is flow of data in sequence.
- The input and output operation between the executing program and the devices like keyboard and monitor are known as “console I/O operation”.
- The input and output operation between the executing program and files are known as “disk I/O operation”.

Classes

- ios :-
  - ios stands for input output stream.
  - This class is the base class for other classes in this class hierarchy.
  - This class contains the necessary facilities that are used by all the other derived classes for input and output operations.
- istream:-
  - istream stands for input stream and handle input stream
  - The extraction operator(>>) is overloaded in this class to handle input streams.
  - This class declares input functions such as get(), getline() and read().

- streambuf :-
  - This class contains a pointer which points to the buffer which is used to manage the input and output streams.
- fstreambase :-
  - It provide operations common to the file streams. Serves as a base for fstream, ifstream and ofstream class.
  - It contains open() and close() function.
- ifstream :-
  - It provides input operations.
  - It contains open() function with default input mode.
  - Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

- ofstream :- (provide output Operations)
  - It contains open() function with default output mode.
  - Inherits the functions put(), write(), seekp() and tellp() functions from the ostream.
- fstream :-
  - This class provides support for simultaneous input and output operations.
  - Inherits all the functions from istream and ostream classes through iostream.
- filebuf :-
  - Its purpose is to set the file buffers to read and write.
  - We can also use file buffer member function to determine the length of the file.

Mode ios::

in *	input	File open for reading: the internal stream buffer supports input operations.
out	output	File open for writing: the internal stream buffer supports output operations.
binary	binary	Operations are performed in binary mode rather than text.
ate	at end	The output position starts at the end of the file.
app	append	All output operations happen at the end of the file, appending to its existing contents.
trunc	truncate	Any contents that existed in the file before it is open are discarded.

tellg and tellp()

tellp()	**tellg()**
It is used with output streams and returns the current “put” position of the pointer in the stream.	This is used with input streams and returns the current “get” position of the pointer in the stream.
Syntax: pos_type tellp();	**Syntax:** pos_type tellg();
It returns the position of the current character in the output stream.	It returns the position of the current character in the input stream.
tellp() gives the position of the put pointer.	tellg() gives the position of the get pointer.

Templates

- It is a simple yet very powerful tool in C++. The simple idea is to pass the data type as a parameter so that we don’t need to write the same code for different data types.
- 2 new keywords added :- templates and type name
  - Type name could be replaced by class keyword

How Templates works??

- Templates are expanded at compiler time.
- Source code contains only function/class, but compiled code may contain multiple copies of the same function/class (for different data-types).

- Both function overloading and templates are examples of polymorphism features of OOP.
- Function overloading is used when multiple functions do quite similar (not identical) operations, templates are used when multiple functions do identical operations.

- Note :- Template Argument Deduction for classes is valid of c++17 or latest only

Template Specialization

- For performing different task for different dataType
- Generic programming is an approach where generic data types are used as parameters in algorithms so that they work for variety of suitable data types.
  Templates are sometimes called parameterized classes or functions.

Using Keyword

Standard Template Library

- It is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc.
- It is a library of container classes, algorithms, and iterators.

.

Containers/classes

- It is a object which holds the data of certain types.
- The container manages the storage space for its elements and provides member functions to access them, either directly or through iterators.
- Sequence Container :-
  - Array (Static Contiguous Memory)
  - Vector (Dynamic Contiguous Memory)
  - Set
  - List (Doubly linked List)
  - Forward-list
  - De-que

Preprocessor

- Same as in C

Namespace

- They provide the extra space such that we can declare variable,function and class (could be of same name)
- Basically provide different space (large scope)
- Namespace doesn’t have specifiers (public private)