For and while: entry control loop

Do while: exit control loop

For: when the no of iterations is given

While: no of iterations not confirmed

~n= -(n+1)

Sizeof() is a operator

>> right shift = divide by 2

<< left shift = multiply by 2

% depends on the sign of Numerator

for(;;) {} -> infinite loop

if 2 conditions given in a loop : 2nd condition is considered

SWITCH:

* WE CAN USE DEFAULT ANYWHERE
* WE CAN USE VALID EXPRESSIONS
* NO DUPLICATE CASE
* CAN’T USE LOGICAL OPERATIONS IN CASE

printf(“%d%d”) -> garbage value

C++ code: 3 approaches

1. object oriented
2. object based
3. procedural

cout -> object of ostream class

<< : insertion operator

Character output stream – to display the output

Cin -> istream class

>> : extraction operator

Character input stream – to take the value from keyboard/user

Default modifier : private

Modifiers: private, public,

getline(cin ,<variable name>) : in order to get a string with space

cin.ignore() : to ignore the nextline

using namespace std:

namespace is a different topic

contains all the built-in functions

DSA : Efficient way of storing data – easy access

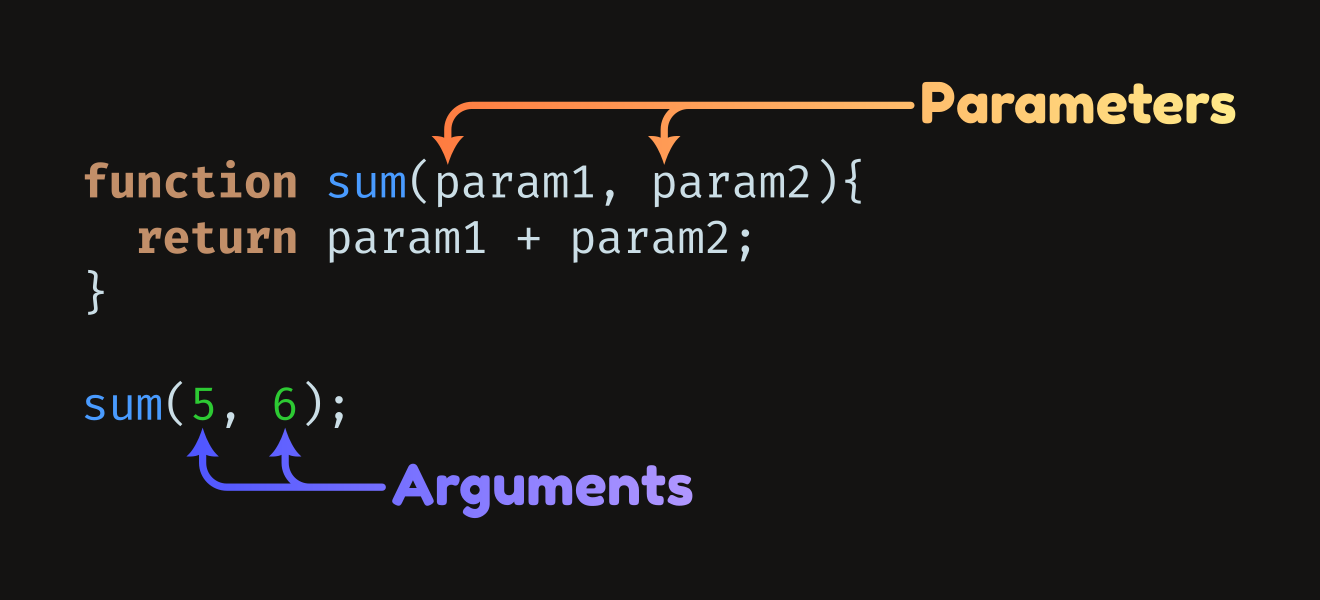
STORAGE CLASSES:

* AUTO
* STATIC
* EXTERN
* REGISTER

Expression: combination of operator and operands

Type casting: converting one type to another type

|  |  |
| --- | --- |
| Int | 4 |
| Float | 4 |
| Char | 1 |
| Double | 8 |
| Long | 4 |
| Long double | 16 |
| Short | 2 |



Types of functions:

User defined

Pre defined

Why do we need user defined function?

We can use them to perform tasks according to our requirements when predefined functions doesn’t fulfil our needs.

Function feature:

* Reusability
* Readability

Polymorphism:

It allows us to use same function in different ways.

Function overloading:

* It’s a compile time polymorphism, which allows us to pass different argument to same function resulting in different outputs based on inputs
* Static polymorphism
* When we may give different argument to same function.

Function overriding:

* It’s a runtime polymorphism, which allows us to modify the definition of a function of a parent class in order to get the desired output.
* Dynamic polymorphism

C++ is a compiled language

Sorting : arranging data in particular order(ascending/descendion)

Bubble sort

Selection sort

STL: Standart Temp Lib

Collection of library/algo, function and other components/data structure to simplify the c++ programming

Conatiners: used to store data in sequential manner to simplify your requirements.

* Sequence container
  + Vector:

It is in dynamic nature

* + Array: predefined function

Advantages: optimization, no need to write complete logic / code is short, data type is decided at runtime

Disadvantages: it is in static nature

* + Deque: double ended
  + list
* Associate container
  + Set
  + Multiset
  + Map
* Container adapter
  + Stack : LIFO

OPERATIONS: push, pop, top ( to see the top), empty

* + Queue: FIFO

Lambda function: are a way to define anonymous function objects, which are useful for short snippets of code that are used only once or for simple operations.

Allows to write inline, unnamed functions within the code.

Helps write function in single line.

[capture](parameters) -> return\_type {

//function body

};

Capture: specifies which variables from the surrounding scope are captured and how

Eg:

auto add= [](int a, int b){ //int won’t work here ☹

return a+b;

}

for\_each(begin, end, function) //will perform the function on all the elements btw start and end

LEET CODE:

STOCK MARKET-SELL AND BUY -I AND -II

PLATFORMS:

GIT LAB

BITBUCKET: CI(continuous integration) /CD(continuous development) OPERATIONS

Software development lifecycle:

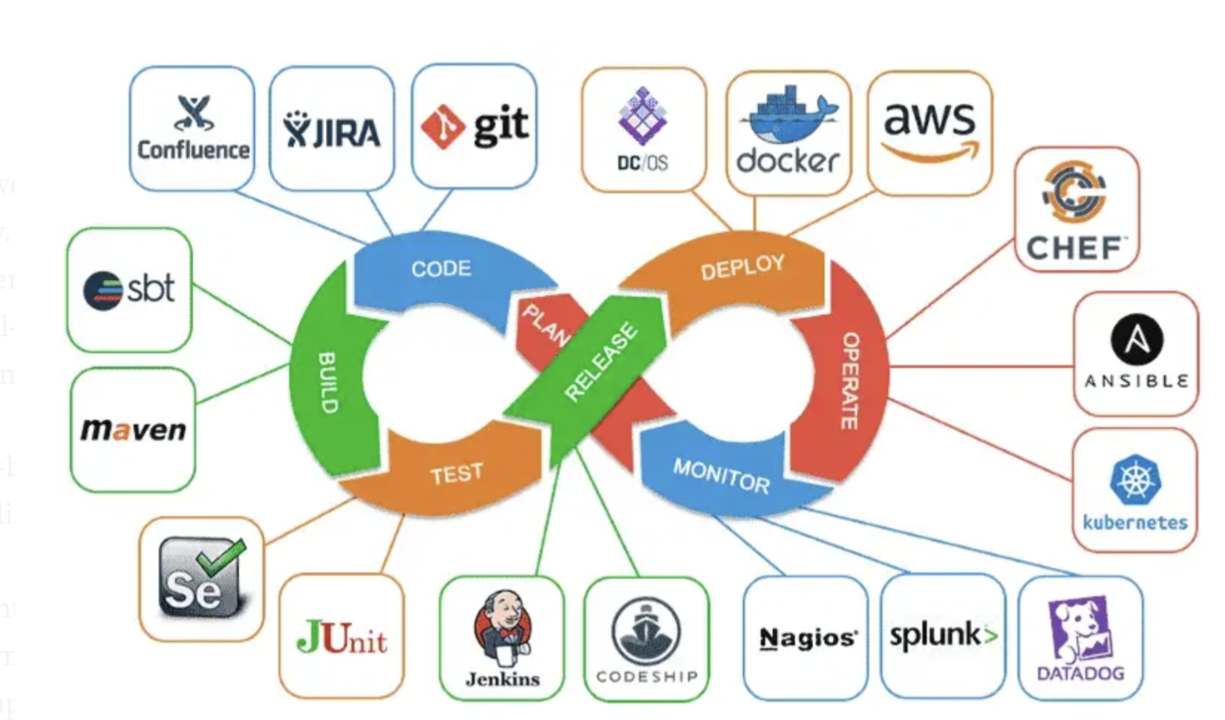
1. planning: info gathering / raw info/ good info
2. definining: good info
3. designing: flow char, test case, class info
4. developing:
5. testing: test case, TDD (Test Driven Development)
6. deploying: put on server 🡪 testing

disadvantages of the traditional software development lifecycle:

1. time consuming
2. costing is more

new approach CI/CD Approch: (Deops engineers: knows all the CI/CD tools)

CI/CD tools includes jenkins, github



ACCCUMULATOR WITH A LAMBDA, LAMBDA WITH MUTABLE, OOPS, POINTERS, ITERATORS

Storage classes:

Auto:

SCOPE: local to the block

LIFETIME: gets destroyed when the block is exited

Register:

SCOPE: local

LIFETIME: destroyed when the block is exited

suggests the compiler to store the variable in CPU register for faster access.

Static: local to the function but retains its value between function calls.

Exists for the duration of the program.

No need to create an object with static member function/data members inside the class.

Extern: Global , the variable or function is visible acroos multiple files.

Exists for the duration of the program

Can use the variable in other file with extern

Mutable: used in classes,as per the containing object, allows a member of an object to be modified even if the object is declared as constant.

Class Example {

Mutable int value;

Public:

Example(int val):value(val){} //constructor

Void modify() const{value++ ;}

}

//dev c++

#std::sort() doesn’t strictly follow the classic quick sort algorithm, it uses a hybrid sorting approach (often introsort) that combines quicksort, heapsort, and insertion sort.

#stable\_sort() : merge sort

Scope Resolution operator

STRUCTURE: collection of variables of different types

Struct (keyword to declare the structure)

Struct Person{

std:: string name;

Int age;

Float height;

}

Person P1;

P1.name;

Each member of structure is stored in a separate memory location

Public private and protected

Sum of size of all members

Can hold multiple member at a time

Safe to access any member at any time.

**In what scenarios will we prefer structure over class??**

Union : same as structure but it provides same memory location for all the members, at a time only one member can be used.

Total size of union ifs size of largest member.

ios\_base::sync\_with\_stdio(false); //stop syncronysation with c

cin.tie(NULL);

cout.tie(NULL);

used to optimise the code