An **array** is a series of memory locations – or 'boxes' – each of which holds a single item of data, but with each box sharing the same name. All data in an array must be of the same data type .

**Vector** containers are implemented as dynamic arrays; Just as regular arrays, vector containers have their elements stored in contiguous storage locations, which means that their elements can be accessed not only using iterators but also using offsets on regular pointers to elements.

But unlike regular arrays, storage in vectors is handled automatically, allowing it to be expanded and contracted as needed.

Can be created by

std::vector<int> items={1,2,3}

items.push\_back(100) //will add 100 to it

items[2] //will allocate to 2 and get it out

items.size() //will show the size of the vector

items[items.size()-1] // will grab the last element of vector dynamically(non stationary)

“\n” goes to the next line

\v vertical tab

\a beeps but not in visualcode

\” adds “

\\ adds a \

Std:boolalpha outputs value in true and false rather than just 1 or 0

Using include<float.h>

Float double and long double give decimals, within this std::fixed fixes the decimal correctness

Const int makes it a constant or use #define X 10(at the start) or enum { y = 100}

Sqrt(34) gives square root

Remainder( 100,3) gives remainder or fmod(100,3)

Fmax(19,3) gives max and Fmin() gives min

And ceil(fmin(19,3.25)) gives higher integer than decimal and floor() gives greatest integer func

Trunc and floor are similar but for say -1.5 trunc gives -1 and floor gives -2

Round will round it off

Using #include <string> and std::string greeting = “hello”;

Greeting += “there” changes the greeting string value for everything to hello there or

Use greeting.append( “ there”)

Greeting.insert(3, “ “) will add any string from after the 3rd position

Greeting.erase(3, 1) erases 1 character after 3rd position

Greeting.pop\_back(); erases the last word

Greeting.replace(1,4, “string”) replaces first 4 letters with string

Greeting.find( “string”) searches for string and can be used in Greeting.replace(greeting.find( “string”),4, “xyz”)

Greeting.find\_first\_of(“string”) will find it for us

Use it with cout<< greeting + “string”

Getline(std::cin, greeting) prints the full user input and cin.getline also allows integers

If(greeting == “string”) std::cout<< “Equals”<<std::endl; will output if equal

== compares value of strings

If(greeting.compare(“hello”) == 0) will compare and for same output it will give value

Auto x =4U will assign x to the category of int UL is long ULL is double long

Auto x=4.6F will assign to F, nothing is double and L is long double

To change number to hexadecimal we add 0x for eg 0x30 which is 48 and for octal only prefix with 0 for eg 030 or we can use cout<< std::hex or std::oct<< number;

To get decimal division we need to add a decimal to one of the numbers to indicate it is a floating integer ie int x = 10.0 / 4

The modulus operator is % which gives the remainder ie int x = 10%3 will give 1 as remainder

CONTROL FLOW INCLUDES BRANCHING AND LOOPING

$ Branching includes if and switch statements

#If(expression) which is T/F and then { //code}

We can also pair this with else if() and else() statements added in the end

The == operator compares two operands and the && is a logical operator and it can be used in if statements ie if(a==b && c==d) will only work if both are right

And the || operator is a logical operator which requires either one to be right to be executed ie

If(a==b || c==d) it requires one of two conditions to be right

If we do not want a particular relation in the if statement we can use !

Ie for eg if( !(a==b)) will only be executed only when they aren’t equal or we can use != in place of ==

#Switch is like if statements but only with int and can be used as

Switch(age)

{ case 1:

Std::cout<< xyz;

Break;

Case 2:……..}

$ Loop has while for and do while

#For loop can be used as

{ for(int i=0;i<10;i++) or for(int i= double;i<10;i++)

{std::cout::……….:

Return 0;}

Where i++ means I begins at 0 and keeps increasing while i-- does the opposite

#While loop can be used as

{ int i=9

While(I > < or = 0)

{ Std::cout<< i << std::endl;

i--;

}

#Dowhile gets the code done atleast once and can be used as

{

do

{

} while(true);

}

WITHIN FOR STATEMENTS

To stop and treat a particular word specially we use break

To remove a particular word we use continue

Conditional Operator is used a

say if points are given for a right answer ie int points = guess== answer ? 10 : 0 then 10 points are given if true and none if they are wrong

CLASSES AND OBJECTS AS WELL AS

FUNCTION, METHOD, STATIC METHOD AND CONSTRUCTOR = TYPES OF FUNCTIONS

An object is created within a class for exampler User abc where User is class and abc is class and static methods are defined for classes(User) for eg User.count() while method is defined for object(xyz) for eg user.speak() which is defined for the particular xyz within User.

Constructor is a function that is called only when a new object is created, in this instance a new User

Also Destructors are called when an object is deleted

NAMESPACES

It includes strings,vectors

HOW TO CONTINUE----------------------

OOP Advanced

Different types of collections

Debugging

Software testing and how to test software

* Regression testing
* Integration testing
* Unit testing

Pointers and dynamic memory

Templatized classes