CS201 Introduction to Programming Lecture Wise Questions and Answers

For Final Term Exam Preparation by Virtualians Social Network

What are macros?

Macros are #define substitutions perform by the preprocessor. #define macros were common in C, but don't have the same prominence in C++ because C++ provides equivalent features which allows the compiler to perform type checking.

How can we define macros?

The #define directive specifies a macro identifier and a replacement list, and terminates with a new-line character. The replacement list, a sequence of preprocessing tokens, is substituted for every subsequent occurrence of that macro identifier in the program text, unless the identifier occurs inside a character constant, a comment, or a literal string. The #undef directive is used to cancel a definition for a macro.

A macro definition is independent of block structure, and is in effect from the #define directive that defines it until either a corresponding #undef directive or the end of the compilation unit is encountered.

The #define directive has the following syntax:

#define identifier replacement-list newline
#define identifier (identifier-list (opt))
Replacement-list newline

If the *replacement-list* is empty, subsequent occurrences of the identifier are deleted from the source file.

The first form of the #define directive is called an *object-like macro*. The second form is called a function-like macro.

Lecture No. 24

What is the Static Memory Allocation in c programming?

Static allocation is what happens when you declare a static or global variable.

What is the Automatic Memory Allocation in c programming?

Automatic allocation happens when you declare an automatic variable, such as a function argument or a local variable.

What is the Dynamic Memory Allocation in c programming?

Dynamic memory allocation is a technique in which programs determine as they are running where to store some information. Dynamic allocation is not supported by C variables but is available via GNU C Library functions.

What is the calloc function in c programming?

calloc is memory allocation function that is normally used for requesting memory space at run time 4 storing data types such as arrays and structures. calloc allocates multiple blocks of storage, each of the same size and then sets all bytes to zero, and then returns a pointer to the memory.

What is the malloc function in c programming?

Malloc is the memory allocation function in C. C requires memory to be manully allocated for some data structures. Malloc will allocate a section of memory of a given size for the data structure. Hope that helps. Sorry I don't have more information off hand. Happy programming.

What is the free () function in c programming?

Free function releases memory that you've allocated and return it to the heap for reallocation.

What is the realloc Function in c programming?

It changes the size of the memory object pointed to by ptr to the size specified by size. The contents of the object will remain unchanged up to the lesser of the new and old sizes. If the new size of the memory object would require movement of the object, the space for the previous instantiation of the object is freed. If the new size is larger, the contents of the newly allocated portion of the object are unspecified. If size is 0 and ptr is not a null pointer, the object pointed to is freed. If the space cannot be allocated, the object remains unchanged

What is the Memory Leak in c programming?

When a computer program acquires memory but fails to release it back to the operating system. In object-oriented programming, a memory leak may happen when an object is stored in memory but cannot be accessed by the running code

What is Dangling Pointers in c programming?

Dangling pointers do not point to a valid object of the appropriate type. These are special cases of memory safety violations.

Lecture No. 25

What is meant by trivial and non-trivial?

Trivial and non-trivial refer to something (a statement) which is easy and not easy to prove. For example it's a trivial fact that a character variable can store only one character at a time.

Get char (); function in programmer

Get char () function takes a single character as input from user. You can store the character into a character variable or an array. For example if you have a character variable

char s;

then s = get char(); statement will take one character from the user and store it in s.

What is Function Overloading

Function overloading means defining multiple functions with same name but with different parameters. For example you may have a function sum() with 2 arguments and with three arguments such as

int sum(int a, int b);

int sum(double a, double b, int c);

What is the difference among calloc function ,malloc function and recalloc function?

1- Two types of dynamic memory allocation one is calloc and other is malloc.

There are two differences. First is in the number of arguments. Malloc() takes a single argument (memory required in bytes), while calloc() needs two arguments.

malloc() does not initialize the memory allocated, while calloc() initializes the allocated memory to ZERO.

calloc() allocates a memory area, the length will be the product of its parameters. calloc fills the memory with ZERO's and returns a pointer to first byte. If it fails to locate enough space it returns a NULL pointer.

Syntax: ptr_var=(cast_type *)calloc(no_of_blocks , size_of_each_block); i.e. ptr_var=(type *)calloc(n,s); malloc() allocates a single block of memory of REQUSTED SIZE and returns a pointer to first byte. If it fails to locate requsted amount of memory it returns a null pointer.

Syntax:

ptr_var=(cast_type *)malloc(Size_in_bytes);
The malloc() function take one argument, which is the number of bytes to allocate, while the calloc() function takes two arguments, one being the number of elements, and the other being the number of bytes to allocate for each of those elements. Also, calloc() initializes the allocated space to zeroes, while malloc() does not.

If a pointer has already been allocated memory using malloc and later this allocated memory needs to be expanded for some reason, the realloc function is used. It will add the consecutive memory spaces to the existing if available; otherwise it will allocate whole new space for the same pointer.

What is Default Function Argument?

Default function argument means to fix the value of one or more than one function parameters which you expect will remain same in the entire or most of the program. For example you have a function named table () which takes two parameters number and upper limit. Number is the value you want to print table of, and upper limit contains the value up to which you want the table to be. In this case you will know that most of the time the value of upper limit will be same i.e. 10, so you can fix this value.
Table (int number, int upper limit = 10);
Now whenever you call this function, the value of upper limit will be 10 by default.
You can also call this function by only giving the parameter value for number variable only.

When a variable is declare static?

Variable declared as static in a function is initialized once, and retains its value between function calls. The default initial value of an uninitialized static variable is zero. If a function or global variable is declared static, it can only be accessed in that file.

Lecture No. 26

Can you give the example of 3 and 4 dimensional arrays. and it is compulsory that for 2 dimensional we have to use nested loops. Can we use 2 dimensional arrays without using even 1 loop? if yes then please give example.

```
main()
{

int a[3][2][2];
int i j k;
 for(i=0;i<3 i++)
for(j=0;j<2;j++)
```

```
for(k=0;k<2;k++)
 {
cout i= dnj= dnk= dnnn i j k;
cout<" enter the value to store at the specified postionn ";
cin>> d a[i][j][k];
}
getch();
}
```

Now turn to your second part of the Question.
Accessing all of the elements of a two-dimensional array requires two loops: one for the row, and one for the column. Since two-dimensional arrays are typically accessed row by row, generally the row index is used as the outer loop.

Is it compulsory that we will initialize outside the main body? if yes the what is the main difference between global variable and it.

I need example that where we can use 3 dimensional rows. and about arrays, if i am making one char array like "char a[10]" so it must accept 9 character but when i am entering the data it is accepting more than 9 like much more as i can enter and even this data is printing. Where this data is storing in the memory while we have limitation to enter just 9 characters.

The const keyword is used to create a read only variable. Once initialised, the value of the variable cannot be changed but can be used just like any other variable. It is not compulsary to intialize them outside the main body.
Example:

```
main()
 {
const int age = 28;
}
```

So here you have delcared in the main () function. One can also declare const out side the main () body.

Question: I did not get the concept of 3d arrays. because if we need 3 columns or 4 or 5, we can do these things using 2d arrays. can you explain conceptually that where we can use 3d arrays mean which

situation? you coded for me 1 example but i did not get. can you explain the purpose and example in real life....

One use of three dimensional array is that when we want to save the record of some person in form of rows and columns and value for each element of the array is a character string. For example, if we want to store the record of VU students that is the student name and student VU ID and we know that both are in form of character strings so doing so can be achieved using a 3D array as in the code given below.

```
#include <iostream.h>
main()
{
   char Student[2][2][10];

   for(int i=0; i<2; i++)
   {
      for(int j=0; j<2; j++)
      {
         for(int k=0; k<10; k++)
         {
            cin>>Student[i][j][k];
         }

      }
   }

   system("pause");
}
```

In the above declared 3D character array, the first index shows number of students whose records will be saved. In this case the first index value is 2 so therefore we are storing the records of 2 students. The second index value is 2 which indicates that for each student, two type of information will be stored that is his/her name and id. The third index value is used for the number of characters for each character string to be stored.

Let spouse we declared one global variable "int a=10;" and later in the main statement we say "a=20;" so now it's value is 10 or 20? please explain in detail and give some examples.....

In such case, the value of the variable will be modified from 10 to 20. Below is the code which explains all this.

```
#include <iostream.h>
int a = 10;
main()
{
      cout"Value of a = "a;
      coutendlendl;

      a = 20;

      cout"New value of a = "a;

      system("pause");
}
```

Compile and run this code and see the program output.

What is directory?

2) Suppose we want to print a string by #define Then how it possible?

1) A directory or in other words a folder is a file system structure in which to store computer files.

2) Below is the code in which we are printing some character string which was declared using #define directive.

```
#include <iostream.h>
#define name "Hello World";
main ()
{

   coutname;

    system("pause");
}
```

A directory or in other words a folder is a file system structure in which to store computer files.

What is a Good Program?

Good program is a generic term and can be defined in many ways but generally we can say that a good program is "a program that performs all required functionalities for which it is made along with efficiency and better memory utilization."

Lecture No. 27

What is difference between malloc and calloc and realloc?

calloc take two argument i.e. calloc(m,n) where first argument is how much space you require in term of number of elements and 2nd argument is space in term of size of each element
For example

calloc(1000,sizeof(int));

Where 1000 is total space that is allocated from heap and it is of size integer.
The main benefit of doing memory allocation through calloc that memory is automatically initialized with 0.

malloc take one argument i.e. malloc(n) where n is number of bytes required.
For example

malloc(1000*(sizeof(int)));

Where again 1000 require space that is allocated from heap and it is multiplied with integer.

If a pointer has already been allocated memory using malloc and later this allocated memory needs to be expanded for some reason, the realloc function is used. It will add the consecutive memory spaces to the existing if available; otherwise it will allocate whole new space for the same pointer.

How to make our own header file with .h extension and then how to include it in another program?

Header files are simply prototype of functions however functions is not implement. As the program gets big, it becomes difficult to write the definitions of all the functions at the beginning of the program. Sometimes, we write the functions in a different file and make the object file. We can include the prototypes of these functions in our program in different manners. One way is to write the prototype of all these functions in the start before writing the program.
The better way is to make a header file and write the prototypes of all the functions and save it as ordinary text file. Now we need to include it in our program using the #include directive. You just create a file with the name xyz.h and add whatever to it and include it in your c++ file using: #include "xyz.h".

What are Utility Functions?

The functions (methods) are normally written in public part of the class. Are there functions which are private to a class? Answer is yes. The functions of a class may be of two categories. One category contains the member functions which manipulate the data or extract the data and display it. Through these, we can set and get values to manipulate data. These are the functions which are in public interface of the class and manipulate the data in the object. But sometimes, we need such functions that is the requirement of these member functions. Suppose we write a setDate function. This function is given an argument and it does the same thing as done by the constructor. In other words, it sets a value of date. Now that function can be public so that it can be called from outside the class. Now we want that the member functions of the class can call this function. But it should not be called from outside. In this case, we put this function in private section of the class. These functions are called utility functions. These are a utility used by other methods of the class. However, they are not functions, supposed to be accessed from outside the class. So they are kept private.

Lecture No. 28

What are constructors?

A constructor is a special member function which is called whenever a new instance of a class is created. The compiler calls the constructor after the new object has been allocated in memory, and converts that "raw" memory into a proper, typed object. The constructor is declared much like a normal member function but it will share the name of the class and it has no return value.

Constructors are responsible for almost all of the run-time setup necessary for the class operation. Its main purpose becomes in general defining the data members upon object instantiation (when an object is declared), they can also have arguments, if the programmer so chooses. If a constructor has arguments, then they should also be added to the declaration of any other object of that class when using the new operator.

What are destructors?

Destructors are usually used to de allocate memory allocated through constructors in the class and do other cleanup for a class object and its class members when the object is destroyed. A destructor is called for a class object when that object passes out of scope or is explicitly deleted. A destructor is a member function with the same name as its class prefixed by a ~ (tilde). For example:

```
class X {
 public:
 // Constructor for class X
 X();
// Destructor for class X
~X();
};
```

What is get line?

getline() is a string function which is used to input/read a string including spaces character.
General syntax of the getline() function is,
getline(name, maxsize);

Where “name” is the name of the array which will store the input/read string and “maxsize” is the size of the array.

Consider the following example,

```
#include <iostream.h>

#include <conio.h>
```

```
#include <string.h>

 main() {

char Arr[100];

cout"Enter a String including space: ";

cin.getline(Arr,100);

cout"The string you entered is: "Arr;

            getche();

           return 0;

}
```

What is class?

A class is a mechanism for creating user-defined data types. It is similar to the C language structure data type. In C, a structure is composed of a set of data members. In C++, a class type is like a C structure, except that a class is composed of a set of data members and a set of operations that can be performed on the class. Once you create a class type, you can declare one or more objects of that class type. For example

```
class X { /* define class members here */
 };
int main() {
 X xobject1; // create an object of class type X X xobject2; // create another object of class type X
}
```

It is user defined data type means that when we need to create a data type of our own choice and we want required operations on that data member than we create a class with the required data

and operations ans use it by creating obects of that class just like we create variables of int , char or double.

Lecture No. 29

What is friend function?

The friend functions of a class have access to the private data members of class. Despite being a good thing, there is possibility of vulnerability. We are opening our thoughts, inside view for somebody else. Without having 100% trusts, it will be risky to make our thoughts and feelings public. We want that our private data is accessible to someone outside, not public for everybody. Otherwise, the data encapsulation and data-hiding concept will be violated. We keep the data members private and declare some specific functions that are not member of the class but friend of the class. As friends, they have access to the inside data structure of the class despite not being members.

How can declare friend functions?

Declaration of Friend functions
to declare a friend function, we can put it anywhere in the class. According to the definition of the friend functions, they have access to the private data members of the class. These can also access the private utility functions of the class. The question arises where we should put the friend function whether in the private or public part of the class. Be sure that friend is a very strong statement. It is too strong to be affected by public or private. We can put it anywhere in the class. But remember that friend functions are not member of the class. So their definition will be always outside the class. However, the prototype of the function will be written in the class. We use the keyword ‘friend’ beforethe prototype of the function.

```
friend    return_type     friend_function_name(int, char);
```

What are friend classes?

Friend Classes

We have seen that a class can define friend functions for itself. Similarly a class can be declared as a friend class of the other class. In that case, the function of a class gets complete access to the data members and functions of the other class. So it is an interesting expansion of the definition that not only the functions but also a class can be a friend of the other class. The syntax of declaring a friend class is that within the class definition, we write the keyword friend with the name of the class. It is going to be a friend class. I.e. friend class-name;
we can also write the word class after the keyword friend and before the class name as.

Lecture No. 30

What's a Dangling Reference?

A dangling pointer is a pointer which points to some random memory location without the knowledge of the coder... usually the after effect of not properly deleting it within the code or because of poor/inexistent pointer assignment error checking. It's called that way because the coder thinks the pointer is referring some memory location while in fact it is not.

It's most dangerous since it's hard to debug. As long as you have your pointers properly deleted as soon as you stop needing them and as long as you always check if a pointer was properly assigned, there should be no problems.

Things get a little bit more complicated when creating classes with dynamic data members (data members that are pointers)... actually these can quickly become a nightmare if not properly implemented. There is a "little" more to these than just error checking and deleting.

2.
A dangling pointer can also be called that way when you declare a pointer but don't assign anything to it. You should always assign something to your pointers. Even if to NULL. As far as memory is concerned, not assigning a pointer is exactly the same thing as 1.

Make it a rule to never have more than 1 statement in which to declare and assign your pointer. This may look a little exaggerated, but discipline is the way to go when memory management is concerned.

What is array?

An array is data structure that stores a collection of individual values that are of the same data type. Arrays are useful because instead of having to separately store related information in different variables (named memory locations), you can store them—as a collection—in just one variable. It is more efficient for a program to access and process the information in an array, than it is to deal with many separate variables.in one dimensional array data is stored in a single dimension while in a two-dimensional array, it is convenient to think of the first subscript as being the row, and the 2nd subscript as being the column. A two dimensional array with 3 rows and 2 columns can be define as follow. int A[3][4] = {{8, 2, 6, 5}, //row 0

{6, 3, 1 ,0},

//row 1

{8, 7, 9, 6}};

//row 2

const keyword is used to make the pointer ro variable constant means that during execution of the program pointer or variable value will not be changed as we have make it constant by using the const keyword.

What is this?

The keyword this identifies a special type of pointer. Suppose that you create an object named x of class A, and class A has a non-static member function f(). If you call the function x.f(), the keyword this in the body of f() stores the address of x. You cannot declare the this pointer or make assignments to it. A static member function does not have this pointer. The type of this pointer for a member function of a class type X, is X* const. If the member function is declared with the const qualifier, the type of the this pointer for that member function for class X, is const X* const. Use the "this" pointer to discover the location of a class instance. Like any pointer, it stores an address. Member function GetAddress() of C++ class X returns the address of a given instance of class X:

```
class X{
 public:
X* const GetAddress() const{
 return this;
};
```

Difference between object and instructor…

A constructor is a special member function which is called whenever a new instance of a class is created. The compiler calls the constructor after the new object has been allocated in memory,

and converts that "raw" memory into a proper, typed object. The constructor is declared much like a normal member function but it will share the name of the class and it has no return value. Constructors are responsible for almost all of the run-time setup necessary for the class operation. Its main purpose becomes in general defining the data members upon object instantiation (when an object is declared), they can also have arguments, if the programmer so chooses. If a constructor has arguments, then they should also be added to the declaration of any other object of that class when using the new operator.

The term 'object', however refers to an actual instance of a class. Every object must belong to a class. Objects are created and eventually destroyed – so they only live in the program for a limited time. While objects are 'living' their properties may also be changed significantly. An example will help clarify what we've said. Suppose we have a class called 'Animal'. Animals have legs, bodies, and brains. This general concept of an Animal does not change. An instance of the Animal class would be a specific animal – like a lion, a cat, or a zebra. These instances of the Animal class would be objects. Whereas the Animal class is a general concept, the instances of that class – the lions, cats, etc. – take that general concept and create a real instance of it. As shown by this example, every object has a lifespan associated with it – a cat or zebra cannot live forever. And, the properties of those objects can change as well while they 'live'; if we have a 'size' variable defined in the class that would of course change as the cat object grows bigger. So we can say that whereas a class is a general concept (like an Animal), an object is a very specific embodiment of that class, with a limited lifespan (like a lion, cat, or a zebra).

What are double colons (::) in c++?

It is called scope-resolution operator. The scope resolution operator (::) is used to define already declared member functions a class outside the class. Also using scope resolution operator helps to differentiate between the simple functions and the member functions of the class. While defining class member functions outside the class, scope resolution operator (::) is used between the class name and the member function name i.e. ship::foo () where ship is a class and foo () is a member function of the class ship. Scope resolution operator is also used to access the static data member of a class using class name.

Lecture No. 31

What is operator overloading?

Operator overloading is the ability to tell the compiler how to perform a certain operation when its corresponding operator is used on one or more variables. For example, the compiler acts differently with regards to the subtraction operator "-" depending on how the operator is being used. When it is placed on the left of a numeric value such as -48, the compiler considers the number a negative value. When used between two integral values, such as 80-712, the compiler applies the subtraction operation. When used between an integer and a double-precision number, such as 558-9.27, the compiler subtracts the left number from the right number; the operation produces a double-precision number. When the - symbol is doubled and placed on one side of a variable, such as --Variable or Variable--, the value of the variable needs to be decremented; in other words, the value 1 shall be subtracted from it. All of these operations work because the subtraction operator "-" has been reconfigured in various classes to act appropriately.

What is Non-member function?

A function which is not a member of class is called non-member function.

For example, consider the following program,

```
#include <iostream.h>

#include <conio.h>

void NonMemberFunction();            // Declearation

class A

{

private:
```

```
public:
        // Member Function of the Class A
void MemberFunction()
{
cout"This is the member function of calss A";
}
};
main()
{
A objA;
objA.MemberFunction();
coutendl;
NonMemberFunction();
   getche();
   return (0);
}
        //Non Member Function
void NonMemberFunction ()
```

{

cout"This is non Member function";

}

Non member function with Special Access:

Non-Member Functions without Special Access

Overloading of the stream I/O operators for the Array class illustrates the use of non-member function to overload an operator when the class in which the operator is defined is unavailable for modification.

Friend Access

Non-member functions may be given special access to the private or protected data of a class for one of two reasons:

- because they need information about the class that is not accessible through the public interface of the class, or
- efficiency considerations make it necessary for this function to bypass the public interface and be given direct access to the class's encapsulated data.

To make this work, an overloading of the stream I/O operators can be written that operate on files (ofstream, ifstream). For output, the following non-member functions is what is needed:

```
ofstream& operator (ofstream& ofs, Array& a) {

ofs.write((char*)(a.array), 20*sizeof(int));

return ofs;

};
```

What is abstract class?

An abstract class is a class that is designed to be specifically used as a base class from which more specific classes can be derived. The object of abstract class type cannot be created a class that contains at least one pure virtual function is considered an abstract class.

Lecture No. 32

What is detail Unary Operators?

A unary operator is an operator that performs its operation on only one operand. On the other hand an operator is referred to as binary if it operates on two operands. Increment (++) and decrement (--) operators are the best examples of unary operators. The increment operator is used to increase the value of a number or variable by 1. The increment operator is performed with the ++ operator. The decrement operator is used to decrease the value of a number or variable by 1. The decrement operator is performed with the -- operator.

What are the easy words the Real Part and Imaginary Part difference?

In FORTRAN, we can say complex x; now x is a variable of type complex and has a real part and an imaginary part. There is no complex data type in C and C++. While trying to solve the quadratic equation on the similar grounds, we may have a complex number as answer i.e. if we have to calculate the square root of -1, an iota (í) will be used. So the combination of real and imaginary parts is called complex number. In C, C++ we deal with such situations with structures. So a structure is not simply a grouping of real world data like students, car etc., it also has mathematical usage like complex number.

The easy words of detail understand. What temp and real between dot operator

```
friend Complex operator + (double d, Complex c)?
{
Complex temp;
temp.real = d + c.real;
temp.imag = c.imag; return temp;
```

Here temp, imag and real are the variable names that are used to represent the real and imaginary part of the complex number as in FORTRAN, we can say

complex x;

now x is a variable of type complex and has a real part and an imaginary part. There is no complex data type in C and C++. While trying to solve the quadratic equation on the similar grounds, we may have a complex number as answer i.e. if we have to calculate the square root of -1, an iota (ί) will be used. So the combination of real and imaginary parts is called complex number. In C, C++ we deal with such situations with structures. A structure is not simply a grouping of real world data like students, car etc., it also has mathematical usage like complex number. So her we use a structure to represent a complex data type.

What is a leap year please clear?

A year, occurring once every four years, that has 366 days including February 29 as an intercalary day. Leap years are added to the calendar to keep it working properly. The 365 days of the annual calendar are meant to match up with the solar year. A solar year is the time it takes the Earth to complete its orbit around the Sun — about one year. But the actual time it takes for the Earth to travel around the Sun is in fact a little longer than that—about 365 ¼ days (365 days, 5 hours, 48 minutes, and 46 seconds, to be precise). So the calendar and the solar year don't completely match—the calendar year is a touch shorter than the solar year.

It may not seem like much of a difference, but after a few years those extra quarter days in the solar year begin to add up. After four years, for example, the four extra quarter days would make the calendar fall behind the solar year by about a day. Over the course of a century, the difference between the solar year and the calendar year would become 25 days! Instead of summer beginning in June, for example, it wouldn't start until nearly a month later, in July. As every kid looking forward to summer vacation knows—calendar or no calendar—that's way too late! So every four years a leap day is added to the calendar to allow it to catch up to the solar year.

What is mean by *this .....we use *this operator in Date class?

We have just seen that a variable which stores a reference to another variable is called a pointer. Pointers are said to "point to" the variable whose reference they store. Using a pointer we can directly access the value stored in the variable which it points to. To do this, we simply have to precede the pointer's identifier with an asterisk (*), which acts as dereference operator and that can be literally translated to "value pointed by". in C++, the "this" keyword is a pointer to the current instance of the class in a function in which the "this" is used. Doing something like:

return (*this);

basically returns the instance of the class, "*" is the dereference operator and "this" is the pointer to the instance.

Lecture No. 33

What is conversion?

In conversion by constructors, described in the previous section, objects of one type can be implicitly converted to a particular class type. This section describes a means by which you can provide explicit conversions from a given class type to another type. Conversion from a class type is often accomplished using conversion functions. Conversion functions use the following syntax:

conversion-function-name:
operator conversion-type-name ()
conversion-type-name:
type-specifier-list ptr-operator$_{opt}$

Elaborate the difference between

char* getCompany(); and char getCompany();

and pros and cons of both

We use pointer to character as a return type of the function because when we declare an array, the array name is actually a pointer to array of characters. Therefore, we declare the getter functions for character array with return type char*

This Operator:
This pointer
Each time your program creates a class instance (e.g. obj1), C++ creates a special pointer called *this*, which contains the address of the current object instance. Each time your program invokes an instance method (e.g. obj1.show()), the compiler pre-assigns a special pointer named *this* to point to the object instance. The value of *this* pointer changes with different instance invocations. C++ recognizes the *this* pointer only when a non-static member of the object instance is executing. The instances, in turn, use this pointer to access the different methods. [3]

Every member function of a class has an implicitly defined constant pointer called *this*. The type of *this* is the type of the class of which the function is member. It is initialized when a member function is called to the address of the class instance for which the function was called.
The following statement is used to return the value to which *this* points to currently.
return (*this);

Lecture No. 34

What is array of objects?

A class is a user-defined data type. Objects are instances of classes the way int variables are instances of ints. Previously, we have worked with arrays of ints. Now, we are going to work with arrays of objects.

What is declaration?

Declarations tell the compiler that a program element or name exists. A declaration introduces one or more names into a program. Declarations can occur more than once in a program. Therefore, classes, structures, enumerated types, and other user-defined types can be declared for each compilation unit.

Definitions specify what code or data the name describes. A name must be declared before it can be used.

What is function prototype?

A function prototype is a declaration in C and C++ of a function, its name,parameters and return type. Unlike a full definition, the prototype terminates in a semi-colon.

```
int getsum(float * value) ;
```

Prototypes are used in header files so that external functions in other files can be called and the compiler can check the parameters during compilation

Why we use the Dynamic Arrays in this language?

Declaring an array with a fixed size like

int a[100000];

has following problems:

1. Choosing a real maximum is often impossible because the programmer has no control over the size of the data sets the user is interested in. Declaring very large arrays can be extremely wasteful of memory, and if there are many such arrays, may prevent the program from running in some systems.

2. Using a small size may be more efficient for the typical data set, but prevents the program from running with larger data sets. If array limits are not checked, large data sets will run over the end of an array with incorrect results. Fixed size arrays can not expand as needed.

These problems can be avoided by dynamically allocating an array of the right size, or reallocating an array when it needs to expand. Both of these are done by declaring an array as a pointer and using the new operator to allocate memory, and delete to free memory that is no longer needed.

Tell about array and function and type of functions variable declaration?

Array:
An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.

Function:
A function is a group of statements that is executed when it is called from some point of the program. The following is its format:
type name (parameter1, parameter2,) {statements}
where:
• Type is the data type specifies of the data returned by the function.
• Name is the identifier by which it will be possible to call the function.
• Parameters (as many as needed): Each parameter consists of a data type specifies followed by an identifier, like any regular variable declaration (for example: int x) and which acts within the function as a regular local variable. They allow passing arguments to the function when it is called. The different parameters are separated by commas.
• Statements are the function's body. It is a block of statements surrounded by braces { }.

There are two types of functions.
1. Built-in functions/pre-defined functions
2. User-defined functions

What are manipulators?

The manipulators are operators used in C++ for organizing output for example we used endl manipulator, set manipulator, ends manipulator etc.

Lecture No. 35

What do input and output really mean?

To get a good idea about what input and output are, think of information as a stream of characters. This makes sense because whatever we enter through a keyboard can only be characters. Suppose the user enters the number 7479....WAIT...! How do you know the user entered a number? The problem is that you don't really know. All you have is a set of 4 characters: '7', '4', '7' and '9'. It is completely up to you, the programmer, whether you want the input to be a number, to be a string, or to be fodder for /dev/random; whether the characters can be valid for the desired type totally depends upon how whether that type can interpret the characters in the input stream as a description for an object of that type.
You have to get the input characters into a recognizable data type for them to be of any use other than as a character array.
An IO stream not only define the relation between a stream of characters and the standard data types but also allows you to define a relationship between a stream of characters and your own classes. It also allows you nearly limitless freedom to manipulate those streams both using object oriented interfaces and working directly on character buffers when necessary. (Of course some of the lower level manipulations may be undefined; for example, you can't probe forward into an input stream to see the future!)

How do streams work?

Streams are serial interfaces to storage, buffers files, or any other storage medium. The difference between storage media is intentionally hidden by the interface; you may not even know what kind of storage you're working with but the interface is exactly the same.

The "serial" nature of streams is a very important element of their interface. You cannot directly make random access random reads or writes in a stream (unlike, say, using an array index to access any value you want) although you can seek to a position in a stream and perform a read at that point.

What are string streams?

Strings are streams and streams are strings. Both are character arrays, but each has a totally different interface (random access strings vs. serial string streams). By providing both std::string and string streams, the C++ standard library ensures that you have the flexibility to choose either interface for your design.
By including the <sstream> header, you can make objects of the istringstream, ostringstream and string stream types. These objects make certain kinds of string manipulations much easier.

What is getter?
A getter is a method that gets the value of a specific property. A setter is a method that sets the value of a specific property. You can define getters and setters on any predefined core object or user-defined object that supports the addition of new properties.

You can also work without getters or setters functions. They are still very useful if you want to perform some action on the data the user is setting/getting prior to updating the class or returning a value; however, for simply setting the input value to a private method of the class these are not necessary but you can use these functions.

Lecture No. 36

What is Stream Manipulations?

After having a thorough look into the properties and object definition of I/O streams, we will discuss their manipulations. Here, there is need of some header files to include in our program the way, cin and couts are used. We include iostream.h and fstream.h while using file manipulations. In case of manipulation of the I/O streams, the header file with the name of iomanip.h is included. It is required to be included whenever there is need of employing manipulators.

What is Stream Manipulators?

Whenever carrying out some formatting, we will want that the streams can manipulate and a number should be displayed in a particular format. We have stream manipulators for doing this. The manipulators

are like something that can be inserted into stream, effecting a change in the behavior. For example, if we have a floating point number, say pi (л), and have written it as float pi = 3.1415926 ; Mow there is need of printing the value of pi up to two decimal places i.e. 3.14 . This is a formatting functionality. For this, we have amanipulator that tells about width and number of decimal points of a number being printed. Some manipulators are parameter less. We simply use the name of the manipulator that works. For example, we have been using endl, which is actually a manipulator, not data. When we write cout endl ; a new line is output besides flushing the buffer. Actually, it manipulates the output stream. Similarly flushwas a manipulator for which we could write cout flush that means flushing the output buffer. So it manipulates the output.

What are Non-Parameterized Manipulators?

Let's start with simple manipulators. We have been dealing with numbers like integers, floats etc for input and output. We know that our number representations are associated with some base. In daily life, the numbers of base 10 are used in arithmetic. When we see 4000 written on a cheque, we understand that it is four thousands written in the decimal number system (base 10). But in the computer world, many systems are used for number representation that includes binary (base 2), octal (base 8), decimal (base 10) and hexadecimal (base 16) systems. A simple justification for the use of these different systems is that computers internally run on bits and bytes. A byte consists of eight bits. Now if we look at the values that can be in eight bits. 256 values (from 0 to 255) can be stored in eight bits. Now consider four bits and think. What is the highest number that we can store in four bits? We know that the highest value in a particular number of bits can be determined by the formula 2n - 1 (where n is the number of bits). So the highest value that can be stored in four bits will be 24 - 1 i.e. 15. Thus the highest value, we can store in four bits is 15 but the number of different values that can be stored will be 2ni.e. 16 including zero. Thus we see that while taking half of a byte i.e. four bits, 16 (which is the base of hexadecimal system) different numbers can be stored in these four bits. It means that there is some relationship between the numbers that have a base of some power of two. So they can easily be manipulated as bit format. Thus four bits are hex. What about eight (octal)? If we have three bits, then it is 23 = 8, which is the base of octal system. Thus, we can use three bits for octal arithmetic.

What is the difference between using square(x) macro and square(x) function?

It is our choice to use either macro or function. Square(x) can is our function or macro which will calculate the square root of a given number. Now the question is that should we use Square(x) macro or Square(x) function? We know function calling is expensive. Because when we call a function, stack is created from the available memory, program state before function calling is saved and resumed from very next line to the function calling and many more reasons.

Dear student, there is more work in function calling and this work can increase if we have excessive function calls in our program. But if we look macro, just like define macro for Square(x), what is happening. In-fact the code written in front of macro name will be substituted at all the places in our program where we are using square macro. See there is no function call, stack creation, control resuming back to our main program and related activates, which we have seen in function calls. So macro is less expensive in the sense that it takes less computational resources. It just substitutes the definition where it is has been called.

What is Template Functions?

A function template is a C++ mechanism that we can use to define a generic function that is type-unspecific. In other words, when we define the function, we do not specify what types are passed to it, nor what types are returned from it, so long as these are types that we wish to replace with specific ones when we write the code that uses the function. However, we may define types for any parameters or return values that are type-specific: such as a generic comparison function that returns a Boolean value of true or false. So, we can define a function, for example, that takes one or more parameters, and only specify their presence, not the types that should be associated with them. A generic example is as follows:

```
template T Maximum (T one, T two)
{
if ( one > two )
return one;
return two;
}
```

Lecture No. 37

What is the function of Destructors?

In object-oriented programming, a destructor is a method which is automatically invoked when the object is destroyed. Its main purpose is to clean up and to free the resources which were acquired by the object.

Over loading insertion and extraction operators….

The stream can be used not only with standard data types nut also with user defined classes. The C++ lets us treat user-defined data types in the same way as that of built in types like int or cahr. To read and display user defined data types like the built in data type, we have to overload the stream extraction (>>) and insertion () operators to accept the user defined data. We can overload the stream the stream operator to read keyboard input and display output for user defined classes. After overloading the stream operator, single input/output statement can be used for input/output of user defined data reducing the number of statements. Without stream operator overloading.

Lecture No. 38

Is it possible to define two functions as given below?
func(int x, int y)
func(int &x, int &y)

You cannot use same function with same number of arguments of same type. However you can use the same function with different number of arguments or at least with different type. If you use function with same number of arguments and same type, compiler will confuse which definition to use so it will show error. And if you use function with different number of arguments, or same number of arguments but of different type then it is fine.

What is the difference between unary and binary operators and how they can be overloaded?

A unary operation is an operation with only one operand, i.e. an operation with a single input, or in other words, a function of one variable.
eg- * Increment: ++x, x++ .A binary operation is a calculation involving two operands.e.g, a+b .It has two operands a and b,so + is a binary operator.

Example for Unary operator:

```
return type class_name :: operator ++ ()
{ // Operator + Function Definition
return ++count;
}
```

A binary operator can be overloaded as a non-static member function with one argument or as a global function with two arguments (one of those arguments must be either a class object or a reference to a class object).

How we increase size of dynamic allocated memory in C?

We have two types of memory allocation in C++; Static and dynamic. Recall, static memory is allocated from stack (a memory area used for this purpose) and examples are;

int c = 15;
double d= 255.255;

On the other hand, Dynamic memory is allocated / de-allocated from / to heap (a special memory area used for this purpose) and examples are;

Dynamic memory allocation using calloc method
int myPtr;
myPtr = (int *) calloc (1000, sizeof (int)) ; // calloc method for dynamic memory allocation and storing result in a pointer. We are allocating 1000 spaces of int data type

Dynamic memory de-allocation using delete operator
delete operator is used to free the dynamic memory when the allocation is done by using new as shown below;

int *iptr; // declaring a pointer
iptr = new int [10]; // Memory for 10 int is allocated dynamically.
delete iptr; // Allocated is freed and returned to the free store.

Now this allocated memory is gone back to heap and available to other functions etc.

In C/C++, the region of memory allocated at runtime is called heap. It is important to note that both Heap is managed by Operating System itself at run time. There may be and surely, there are many parallel processes for operating system, user processes etc. are accessing memory dynamically. So the heap size is continuously varying.

Lecture No. 39

What is a copy constructor?

A copy constructor is a special constructor for a class/struct that is used to make a copy of an existing instance. According to the C++ standard, the copy constructor for MyClass must have one of the following signatures:

1

2

3

4

MyClass(MyClass& other); MyClass(MyClass& other); MyClass(MyClass& other); MyClass(MyClass& other);
Note that none of the following constructors, despite the fact that they could do the same thing as a copy constructor, are copy constructors:
1

2

MyClass(MyClass* other); MyClass(MyClass* other);
Or my personal favorite way to create an infinite loop in C++:
MyClass(MyClass other);

When do I need to write a copy constructor?

First, you should understand that if you do not declare a copy constructor, the compiler gives you one implicitly. The implicit copy constructor does a member-wise copy of the source object. For example, given the class:
1

2

3

4

5

MyClass { x; c; std::string s; };

The compiler-provided copy constructor is exactly equivalent to:

1

2

3

MyClass::MyClass(MyClass& other) : x(other.x), c(other.c), s(other.s) {}

In many cases, this is sufficient. However, there are certain circumstances where the member-wise copy version is not good enough. By far, the most common reason the default copy constructor is not sufficient is because the object contains raw pointers and you need to take a "deep" copy of the pointer. That is, you don't want to copy the pointer itself; rather you want to copy what the pointer points to. Why do you need to take "deep" copies? This is typically because the instance owns the pointer; that is, the instance is responsible for calling delete on the pointer at some point (probably the destructor). If two objects end up calling delete on the same non-NULL pointer, heap corruption results.

What is the practical implication of all this? Consider the following:

Many of the STL containers and algorithms require that an object be copyable. Typically, this means that you need to have the copy constructor that takes a const reference, for the above reasons.

What is an assignment operator?

The assignment operator for a class is what allows you to use = to assign one instance to another. For example:

There are actually several different signatures that an assignment operator can have:

(1) MyClass& operator=(const MyClass& rhs);
(2) MyClass& operator=(MyClass& rhs);
(3) MyClass& operator=(MyClass rhs);
(4) const MyClass& operator=(const MyClass& rhs);
(5) const MyClass& operator=(MyClass& rhs);
(6) const MyClass& operator=(MyClass rhs);
(7) MyClass operator=(const MyClass& rhs);
(8) MyClass operator=(MyClass& rhs);
(9) MyClass operator=(MyClass rhs);

When do I need to write an assignment operator?

First, you should understand that if you do not declare an assignment operator, the compiler gives you one implicitly. The implicit assignment operator does member-wise assignment of each data member from the source object. For example, using the class above, the compiler-provided assignment operator is exactly equivalent to:

In general, any time you need to write your own custom copy constructor, you also need to write a custom assignment operator.

What is meant by Exception Safe code?

A little interlude to talk about exception safety, because programmers often misunderstand exception handling to be exception safety.

A function which modifies some "global" state (for example, a reference parameter, or a member function that modifies the data members of its instance) is said to be exception safe if it leaves the global state well-defined in the event of an exception that is thrown at any point during the function.

How do I write an exception safe assignment operator?

The recommended way to write an exception safe assignment operator is via the copy-swap idiom. What is the copy-swap idiom? Simply put, it is a two-step algorithm: first make a copy, then swap with the copy. Here is our exception safe version of operator=:

Here's where the difference between exception handling and exception safety is important: we haven't prevented an exception from occurring; indeed, the copy construction of tmp from rhs may throw since it will copy T's. But, if the copy construction does throw, notice how the state of *this has not changed, meaning that in the face of an exception, we can guarantee that *this is still coherent, and furthermore, we can even say that it is left unchanged.

Lecture No. 40

What is NEW Operator?

New is an operator, a keyword which is reserved by C/C++ language for dynamic memory allocation in classes. We can't use new for variable / object names. New is not a function or method name.

Whenever we use new operator for dynamically allocating memory (from heap), following will be happened in sequence;

1. New operator will automatically determine the required memory size of object.

2. Then new operator will call the constructor of the class

3. Finally, it will returns pointer of the class type.

C++ Class Definitions:

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations. For example we defined the Box data type using the keyword class as follows:

class Box { public: double length; // Length of a box double breadth; // Breadth of a box double height; // Height of a box };
The keyword public determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as private or protected which we will discuss in a sub-section.

Define C++ Objects:

A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types. Following statements declare two objects of class Box:

Box Box1; // Declare Box1 of type Box Box Box2; // Declare Box2 of type Box
Both of the objects Box1 and Box2 will have their own copy of data members.

Lecture No. 41

What are Template Functions?

Here are two different types of templates in C++ language i.e.' function templates and class templates. Before going ahead, it will be sagacious to know what a template is? You have used a lot of templates in the childhood. There are small scales being marketed at the stationary shops having some figures on them like circle, a square, rectangle or a triangle. We have been using these articles to draw these shapes on the paper.

We put the scale on the paper and draw the lines with the pencil over that figure to get that shape. These engraved shapes are generally called stencils. But in a way, these are also templates. We may also take these 'cut-outs' as sketches. So a template is a sketch to draw some shape or figure. While drawing a special design, say of furniture, we develop a template for this, which is not an actual piece of furniture.

What is Overloading Template Functions?

Under the techniques employed in function overloading, the functions have the same name but differ either by the number of arguments or the type of the arguments.

Remember that:

The return type is not a differentiator when you are overloading the functions. Now if the number or type of the arguments is different and the function name is same, the compiler will automatically call the correct version. The same rule applies to the template function.

We can write overloaded template functions as long as there is use of different number or type of arguments. We have written a templatized swap function. Let's rename that function as inverse. It will swap the variables.

We have another inverse function that takes one argument and return the minus of the argument supplied. We have two template functions named inverse.

What happens when we say 2 + a;?

In case of ordinary integers, 2 + 3 is same as 3 + 2. Sowed want the same behavior in a class. We want 2 + a behaving the same way as a+ 2.

We cannot carry out overloading of a member function for this. When we write 2 + a; there will be an int on left- hand side of the + operator. It is not a member function of the class. For a member function or member operator, the object on left- hand side

should be that of the class. So if we want 2 + a ; we have to write a friend function for it.

What is Standard Template Library (STL)?

Suppose that the size of array is 100. We want to add the 101st element in the array. We can do it by copying the same array in a new big array and adding the element to that array. Thus we have solutions for different problems, but these are the things of very common use. Their everyday use is so important that two researchers wrote a whole library of common use functions. This library is a part of the official standard of C++. It is called STL i.e. Standard Template Library. As a library, it is a tested code base.

What is Object Oriented Programming?

This type of programming uses sections in a program to perform certain tasks. It splits the program into objects that can be reused into other programs. They are small programs that can be used in other software. Each object or module has the data and the instruction of what to do with the data in it. This can be reused in other software directly.

What is Structured programming?

It splits the tasks into modular forms. This makes the program simpler and easier to read with less lines and codes. This type of program accomplishes certain tasks for that a specific reason. For example, invoice printers use structured programming. This type has clear, correct, precise descriptions.

Lecture No. 42

What is class Templates?

Creation of a data type of our own with the same behavior for int, float and double etc. is the case of defining a complete interface and implementation in a generic fashion. To further understand this concept, let's talk about a data structure called stack.
You might have seen the plates, kept together in a orderly manner i.e. one on the other. This is a stack.

What is the used of templates and friend functions?

Template allows the functions and classes to be defined with any type. Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

Friend function allows access to private or protected data in a class from outside the class. It is function that is used for accessing the non-public members of a class.

What is pop and push function?

These two functions are just like unshift and shift, except they add or delete from the right side of an array (the last position). So, if you want to add an element to the end of an array, you would use the push function. Similarly if you want to unshift an element, then you will use pop function.

Lecture No. 43

What are Matrices in programming language?

Matrices are a concept of Mathematics. Your question is not clear. You may implement a class Matrix. As in math's, you apply different functions on matrices; same you can do in class. You can perform arithmetic functions on matrices. You can take transpose etc. You have to define a 2x2 dimensional array of integer or of any dimension which you want.

What are manipulators?

In C/C++, sometimes we need our input and output to be formatted. For this purpose, we have Manipulators in C/C++ language which changes the behavior of data in stream. If we want to output the data to display in some particular format, we will use these stream manipulators.

We are using streams in almost every program. These are cout (output stream) and cin (input stream). You can think Streams like doors. One for input and other for output. cin is a door which allows data to come in our program and cout for data to go out form our program and print to the screen.

setprecision() is a Manipulator in C/C++ which takes parameter. Suppose we are doing floating point calculation in our program and we want the result to be shown 2 points after the decimal. We will write;

cout setpresicion(2) result;

The output will limit the number with one decimal place. Remember to use 'fixed' keyword with the setprecision.

Sample code for setprecision(2)

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
main()
{

float f = 10.123;

cout "Value of f is" f endl;

cout "\nValue f using Setprecision is " fixed setprecision(1) f ;

getche();

}
```

The output of the program will be;

Value of f is 10.123

Value of f using Setprecision is 10.1

Do not forget to include iomanip.h header file in your program while manipulating with I./O streams. iomaniop is short hand for input output manipulation. Remember, we are including iostream.h since our very first program.

Lecture No. 44

What is Matrix class?

The data structure of the Matrix class is very simple. We have defined an arbitrary number of functions and operators. You may add and subtract more functions in it.
Let's discuss the code beginning with the data structure of the class. In keeping theconcepts of data hiding and encapsulation, we have put the data in the private section of the class. We have defined number of rows (i.e. numRows) and number of columns (i.e. numCols) as integers. These will always be whole number. As we cannot have one and a half row or column, so these are integers.
The private part of the Matrix class is:

```
int numRows, numCols;

double **elements;
```

What is the definition of Matrix Constructor?

Let's start with the default constructor. Its prototype is as under:
Matrix(int = 0, int = 0); // default constructor
We are using the default argument values here. In the definition of this function, we will not repeat the default values. Default values are given at one place. The definition code is as:

```
Matrix::Matrix(int row, int col) //default constructor
{
numRows = row;
numCols = col;
elements = new (double *) [numRows];
for (int i = 0; i < numRows; i++){
elements[i] = new double [ numCols];
for(int j = 0; j < numCols; j++)
elements[i][j] =0.0; // Initialize to zero
}
}
```

Two integers are passed to it. One represents the number of rows while the other is related to the number of columns of the Matrix object that we want to create..
We have declared elements as **elements i.e. the array of pointers to double. We can directly allocate it by getting the elements of numRows times numCols of type double from free store. But we don't have the double pointer.

What is Destructor of Matrix Class?

'Destructor' is relatively simple. It becomes necessary after the use of new in the constructor.
While creating objects, a programmer gets memory from the free store. So in the destructor, we have to return it. We will do it as:

```
delete [] elements;
```

Remember that 'elements' is the variable where the memory has been allocated. The [] simply, indicates that it is an array.

What are Utility Functions of Matrix?

The functions getRows() and getCols() are relatively simple. They do not change anything in the object but only read from the object. Therefore we have made this function constant by writing the const keyword in the end. It means that it does not change anything. The code of the getRows() functions is as follows:

```
int Matrix :: getRows ( ) const
{
return numRows;
}
```

This function returns an int representing the number of rows. It will be used in the main function as

```
i = m.getRows();
```

Where i is an int and m is a Matrix object. Same thing applies to the getCols() function. It is of type const and returns an int representing the number of columns.

What is the difference between object and class?

We know that C++ is an object-oriented programming language. What does it actually mean?

It means we can create classes, make their objects and manipulate them. Classes provide us capability to make our own custom build data types. For example, we can make class of Date. After making class we need to make its instances, usually called as objects. These objects have the capability of what we have defined in the class.

We create class typically containing our public and private data members along with their getter / setter methods. Then we create instance of our class. E.g. name of our class is "Course" and the instance or object we are going to instantiate is CS201. We will write as;

Lecture No. 45

What is Assignment Operator Function?

This operator occupies very important place in the field of programming. When we want to write code like a = b; where a and b both are matrices, the assignment operator attains a critical role as our class does dynamic memory allocation.
At first, we look at the declaration line of the assignment operator, it is written below. The declaration line states that it must return a reference to a matrix.
While dealing with the code of the assignment operator, we will first check whether it is for the self-assignment or not.
To ascertain it, we will write the following line.

```
if( &m != this)
```

What are the Loops and Decisions?

The loops and decisions are 'bread and butter' for a programmer.
While talking about decisions, we read that in C and C++ languages, there is a simple if statement.
There is also 'if-else statement'. We can write a big structure by using the nested if-else statements. There is also switch statement. These statements (if, if-else and switch) are language specific.
Almost all the modern programming languages provide a decision structure.
The exact syntax of which can be got from language reference.

What is Structured Query Language?

In the business world, most of the programming is database-oriented. In today's databases, like Oracle and SQL Server, a different kind of language is used. These are the languages that are called as structured query languages i.e. SQL. SQL is so important that a standard has been developed for it. So there is an ANSI standard for this language.

In the business world, most of the programming is database-oriented. In today's databases, like Oracle and SQL Server, a different kind of language is used. These are the languages that are called as structured query languages i.e. SQL. SQL is so important that a standard has been developed for it. So there is an ANSI standard for this language.