Experiment No.1

1. **Title**: Develop a Student Database Program.

# Problem Statement:

Develop a program in C++ to create a database of student information system containing the following information: Name, Roll number, Class, Date of Birth, Blood group, Contact address, telephone number, marks of 3 subject,etc Construct the database with suitable member functions.Make use of Default constructor, Copy constructor, Destructor, Static member functions, Friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

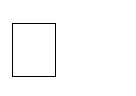
# Objectives:

Students will able to understand the concept of constructor, default constructor, copy, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

# Outcomes:

* + To understand various kinds of Constructor.
  + To understand various concepts of the C++ programming like this pointer, inline code, friend class, dynamic memory allocation, etc.
  + To apply and test various concepts of the C++ programming like this pointer, inline code, friend class, dynamic memory allocation, etc.

# Software/Hardware/Other Requirements:

* + Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more.
  + Operating System – ubuntu/Fedora 64bit OS  Software: G++ compiler/ GCC compiler, Code Editor

# Theory:

1. **Constructors**:
   * A constructor function is a special member function that is a member of a class and has the same name as that class, used to create, and initialize objects of the class.
   * To create a constructor, use the same name as the class, followed by parentheses ().
   * Characteristics
   1. These are called automatically when the objects are created.
   2. All objects of the class having a constructor are initialized before some use.
   3. These should be declared in the public section for availability to all the functions.
   4. Return type (not even void) cannot be specified for constructors.
   5. These cannot be inherited, but a derived class can call the base class constructor.
   6. These cannot be static.
   7. Default and copy constructors are generated by the compiler wherever required.
   8. These can have default arguments as other C++ functions.
   9. A constructor can call member functions of its class.
   10. We can use a constructor to create new objects of its class and constructors cannot be virtual.
   * Syntax:

class class\_name

{

public:

class\_name();

};

| P a g e Army Institute of Technology, Pune

* + Constructors can be classified into 3 types

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor
4. Default Constructor: Default Constructor is also called as Empty Constructor which has no arguments and It is Automatically called when we create the object of class but Remember name of Constructor is same as name of class and Constructor never declared with the help of Return Type. Means we can’t declare a Constructor with the help of void Return Type., if we never Pass or declare any Arguments then this called as the Copy Constructors.
5. Parameterized Constructor: This is another type constructor which has some Arguments and same name as class name but it uses some Arguments So For this, we have to create object of Class by passing some Arguments at the time of creating object with the name of class. When we pass some Arguments to the Constructor then this will automatically pass the Arguments to the Constructor and the values will retrieve by the Respective Data Members of the Class.
6. Copy Constructor: In this Constructor we pass the object of class into the Another Object of Same Class. As name Suggests you Copy, means Copy the values of one Object into the another Object of Class .This is used for Copying the values of class object into an another object of class So we call them as Copy Constructor and For Copying the values We have to pass the name of object whose values we wants to Copying and When we are using or passing an Object to a Constructor then we must have to use the & Ampersand or Address Operator.

# Destructors

* + A destructor is a special function that is a member of a class and has the same name as that class used to destroy the objects.
  + A destructor is automatically called when:
  1. The program finished execution.
  2. When a scope (the { } parenthesis) containing local variable ends.
  3. When you call the delete operator.
  + Destructor Rules

1. Name should begin with tilde sign (~) and must match class name.
2. There cannot be more than one destructor in a class.
3. Destructors do not allow any parameter.
4. They do not have any return type, just like constructors.
5. When you do not specify any destructor in a class, compiler generates a default destructor and inserts it into your code.
   * Syntax:

class class\_name

{

public:

};

~class\_name();

# Static Data Members

* + Static data members of a class are also known as “class variables”. Because their content does not depend on any object.
  + They have only one unique value for all the objects of that same class.
  + Static variables are initialized to zero before the first object is created.
  + Static members have the same properties as global variables but they enjoy class scope.

# Static Member Functions

* + Like static member variable, we can also have static member functions. Member functions that are declared with static specifier.
  + Syntax:

class class\_name

{

public:

};

static ret\_dt fun\_name(formal parameters);

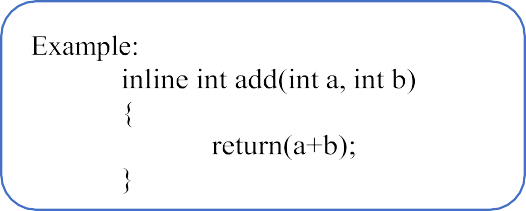
* + Special features:
  1. They can directly refer to static members of the class.
  2. They can be called using the class name (instead of its objects) as: Class name::function name;
  + Limitation: They can have access to only other static members (functions or variables) declared in the same class.

C++ Program to demonstration of static members

|  |
| --- |
| class test {  int code; static int count;  public:void setcode(void) {  code=++count;  }  void showcode(void){  cout<<"\n Object no.:"<<code;  }  static void showcount(void){ cout<<"\n count:"<<count;  }  };  int test::count; int main() {  test t1,t2; t1.setcode(); test::showcount(); t2.setcode(); test::showcount(); t1.showcode(); t2.showcode(); return 0;  } |
| Output: count:1 count:2 Object no.:1  Object no.:2 |

# Inline Functions/Code

* + The inline functions are a C++ enhancement feature to increase the execution time of a program. Functions can be instructed to compiler to make them inline so that compiler can replace those function definitions wherever those are being called.
  + Conditions to use inline function:
  1. Should be small for better efficiency.
  2. Increases the efficiency but all functions should not be inline.
  3. Does not work where switch and goto exists.
  4. Does not use static variables. 5) It cannot be recursive.





Syntax:

# Friend Function & Class

* + In principle, private and protected members of a class cannot be accessed from outside the same class in which they are declared. However, this rule does not affect friends.
  + Friends are functions or classes declared as such. If we want to declare an external function as friend of a class, thus allowing this function to have access to the private and protected members of this class, we do it by declaring a prototype of this external function within the class, and preceding it with the keyword friend.
  + A friend class can access all the private and protected members of other class. In order to access the private and protected members of a class into friend class we must pass on object of a class to the member functions of friend class.
  + For example: class B; class A { friend class B; // class B is a friend class of class A ... .. ...

};

class B { };

* + All member functions of class B will be friend functions of class A.
  + Thus, any member function of class B can access the private and protected data of class A. But, member functions of class A cannot access the data of class B.

|  |
| --- |
| C++ Program to demonstration of friend class |
| #include<iostream> using namespace std; class Rectangle  {  int L,B;  public: Rectangle()  {  L=10; B=20; |
| }  friend class Square;  };  class Square  {  int S;  public: Square()  {  S=5;  }  void Display(Rectangle &Rect)  { cout<<"\n\n\tLength : "<<Rect.L; cout<<"\n\n\tBreadth : "<<Rect.B; cout<<"\n\n\tSide : "<<S; }  };  int main()  {  Rectangle R; Square S;  S.Display(R); //Statement 2 } |
| Output:  Length : 10  Breadth : 20  Side : 5 |

# Pointers:

* + A pointer is a derived data type that refers to another data variable by storing the variables memory address rather than data.
  + Declaration of pointer variable is in the following form: Data\_type \* ptr\_var;

Example: int \*ptr;

* + Here ptr is a pointer variable and points to an integer data type. We can initialize pointer variable as follows int a, \*ptr; // declaration ptr = &a //initialization

# Pointers to objects:

* + Consider the following eg item X; // where item is class and X is object
  + Similarly, we can define a pointer it\_ptr of type item as follows Item

\*it\_ptr ;

* + Object pointers are useful in creating objects at runtime. We can also access public members of the class using pointers.
  + Example: item X; item \*ptr = &X;
  + The pointer ‘ptr ‘is initialized with address of X.
  + we can access the member functions and data using pointers as follows ptr->getdata(); ptr->show();

# This Pointer

* + C++ uses a unique keyword called this to represent an object that invokes a member function.
  + this is a pointer that points to the object for which this function was called. This unique pointer is automatically passed to a member function when it is called.
  + Important notes on this pointer:
  1. this pointer stores the address of the class instance, to enable pointer access of the members to the member functions of the class.
  2. this pointer is not counted for calculating the size of the object.
  3. this pointer is not accessible for static member functions. 4) this pointer is not modifiable.

|  |
| --- |
| C++ Program to demonstration of this pointer |

|  |
| --- |
| class Demo  {  private: int num;  char ch;  public: void setValues(int num, char ch) { this->num =num;  this->ch=ch;  }  void displayValues() { cout<<num<<endl; cout<<ch;  }  };  int main()  {  Demo obj; obj.setValues(100, 'A’);  obj.displayValues(); return 0;  } |
| Output: 100  A |

# Dynamic Memory Allocation: new operator

* + The program to create our own variables “on the fly” is called dynamic memory allocation and is only possible through the use of pointers.
  + To dynamically allocate memory means that a program, while running, this is done through new operator.
  + The operand of the new operator is the data type of the variable being created.
  + Syntax of new operator:

pointer\_variable = new datatype;

* + Syntax to initialize the memory:

\*iptr = new datatype(value);

* + A more practical use of the new operator is to dynamically create an array.
  + Syntax to allocate a block of memory/ dynamic Array,

\*iptr = new datatype[size];

# Dynamic Memory De-allocation: delete operator

* + When a program is finished using a dynamically allocated chunk of memory, it should release it for future use.
  + The delete operator is used to free memory that was allocated with new.
  + Here is an example of how delete is used to free a single variable, pointed to by iptr: delete iptr;
  + If iptr points to a dynamically allocated array, the [] symbol must be placed between delete and iptr:

delete [] iptr;

* + Failure to release dynamically allocated memory can cause a program to have a memory leak.

# Algorithm for the problem Statement:

1. Begin
2. Enter Student information such as Name, Date of Birth, Blood group, class, Contact address, telephone number, 3 subjects marks with help of class instance and methods.
3. Create another class instance and copy first instance content into another with help of Copy Constructor.
4. Print ‘MENU’.
   1. Print ‘1. Insert’.
   2. Print ‘2. Display All’.
   3. Print ‘3. Search Particular’.
   4. Print ‘4. Count Number of Students’.
   5. Print ‘5. Delete Specific’.
   6. Print ‘6. Exit’.
5. Print ‘Enter your choice character’.
6. If op==’6’ then goto step 8 otherwise follow the below steps
7. Switch(op)
   1. case 1:
      1. Read Student information such as Name, Date of Birth, Blood group, class, Contact address, telephone number, 3 subjects marks with help of class instance and methods. ii. break
   2. case 2:
      1. Print all information from database with help of class instance & methods.
      2. break
   3. case 3:
      1. Read Search Student Roll number. ii. For i=0 to n-1:
         1. If Search\_key==st[i].rollno:
            1. Print ‘Search key found at i+1 Location.’
            2. break;
         2. Else: Continue;

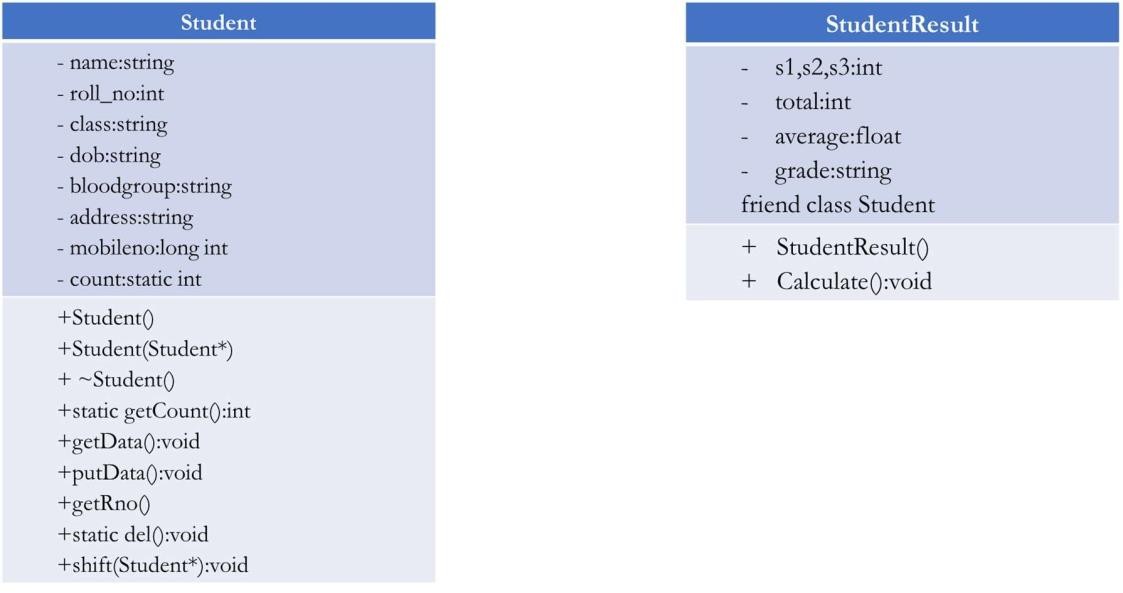
iii. break

* 1. case 4:
     1. Print Student count with help of static member function getStudCount. ii. break
  2. case 5:
     1. Read Delete Student Roll number.
     2. For i=0 to n-1:
        1. If delete\_key==st[i].rollno:
           1. flag=1
           2. Break for loop;
        2. Else: Continue;
     3. If flag=0: print ‘Record not present.’ iv. Else:
        1. Shift all Records left side.
        2. Decrease static student counter by 1.

v. break

* 1. default: iii. Print ‘Invalid Option’. iv. break

1. while(1) then goto step 3.
2. Stop.
3. Flowchart/Class Diagram



1. Implementation:

|  |
| --- |
| C++ Program for Student Database |
| #include<iostream> #include<iomanip> #include<string.h> using namespace std; class StudentResult  { int s1,s2,s3,total; float avg;  string grade; public:  StudentResult()  {  s1=s2=s3=total=0; avg=0.0f; grade="";  } |

void Calculate()

{

total=s1+s2+s3; avg=total/3; if(avg>=66.66f)

grade="First with Dist class"; else if(avg>=60.00f && avg<66.66f)

grade="First class";

else if(avg>=50.00f && avg<60.00f) grade="Second class";

else if(avg>=40.00f && avg<50.00f) grade="Pass class";

else

}

grade="Fail class";

friend class Student;

}; class Student

{ int rno;

string name,cls,dob,bloodGroup,address; long long int mobileno;

static int studCnt; StudentResult \*sr;

public:

Student() //Default Constructor

{ rno=mobileno=0; name=cls=bloodGroup=""; dob="dd/mm/yy";

sr=new StudentResult;

}

static int getStudentCount()

{

return studCnt ;

}

Student(Student \*stud) //Copy Constructor

{

this->rno=stud->rno; this->name=stud->name; this->cls=stud->cls;

this->dob=stud->dob;

this->bloodGroup=stud->bloodGroup; this->address=stud->address;

this->mobileno=stud->mobileno; this->sr->s1=stud->sr->s1;

SE-Comp-A Object Oriented Programming Lab

this->sr->s2=stud->sr->s2; this->sr->s3=stud->sr->s3;

this->sr->total=stud->sr->total; this->sr->avg=stud->sr->avg; this->sr->grade=stud->sr->grade;

}

void getData()

{

cout<<"\n\t Enter Student Persoanal Details:"; cout<<"\n Roll Number:";

cin>>rno; cout<<"\n Name:";

cin.get(); getline(cin,name);

cout<<"\n Class:";

fflush(stdin);

cin>>cls;

cout<<"\n Date of Birth (DD/MM/YY):"; cin.get();

cin>>dob;

cout<<"\n Blood Group:"; cin.get();

cin>>bloodGroup; cout<<"\n Address:";

cin.get(); getline(cin,address);

cout<<"\n Mobile Number:"; cin>>mobileno;

cout<<"\n\t Enter Student Result Data:"; cout<<"\n Subject 1 Marks:";

cin>>sr->s1;

cout<<"\n Subject 2 Marks:"; cin>>sr->s2;

cout<<"\n Subject 3 Marks:"; cin>>sr->s3;

studCnt++;

}

void putData()

{

cout<<setw(5)<<rno<<left<<setw(20)<<left<<name

<<setw(6)<<left<<cls<<setw(11)<<left<<dob

<<setw(7)<<left<<bloodGroup<<setw(25)

<<left<<address<<setw(11)<<left<<mobileno; sr->Calculate();

SE-Comp-A Object Oriented Programming Lab

cout<<setw(4)<<sr->s1<<setw(4)<<sr->s2<<setw(4)<<sr->s3

<<setw(6)<<sr->total<<setw(8)<<setprecision(5)<<sr->avg

<<setw(20)<<sr->grade<<endl;

}

inline int getRno()

{

return rno;

}

~Student()

{

cout<<"\n Exiting from the Class.";

}

void shift(Student \*stud)

{

this->rno=stud->rno; this->name=stud->name;

this->cls=stud->cls; this->dob=stud->dob;

this->bloodGroup=stud->bloodGroup; this->address=stud->address;

this->mobileno=stud->mobileno; this->sr->s1=stud->sr->s1;

this->sr->s2=stud->sr->s2; this->sr->s3=stud->sr->s3;

this->sr->total=stud->sr->total; this->sr->avg=stud->sr->avg;

this->sr->grade=stud->sr->grade;

}

static void del()

{

studCnt--;

}

};

int Student::studCnt; int main()

{

{

Student s1; s1.getData();

cout<<"\n \n"; cout<<setw(5)<<left<<"R No"<<setw(20)<<left<<"Name"<<setw(6)<<left

<<"Class"<<setw(11)<<left<<"Birth Date"<<setw(7)<<left<<"BGroup"

<<setw(25)<<left<<"Address"<<setw(11)<<left<<"Mobile No"

SE-Comp-A Object Oriented Programming Lab

<<setw(4)<<left<<"S1"<<setw(4)<<left<<"S2"<<setw(4)<<left<<"S3"

<<setw(6)<<left<<"Total"<<setw(8)<<left<<"Average"<<setw(20)<<left

<<"Grade"<<endl; s1.putData();

Student s2(s1); s2.putData();

cout<<"\n \n";

}

int cnt,i,choice,key,flag=0; char ch; cout<<"\n\t Max Number of Student into Class:"; cin>>cnt; Student \*st=new Student[cnt]; do

{

cout<<"Student Database Program"; cout<<"\n 1. Add New Student"; cout<<"\n 2. Show all Students"; cout<<"\n 3.

Search Particular Student"; cout<<"\n 4. How many Students Data Entry was Done?"; cout<<"\n 5. Delete Student";

cout<<"\n Enter your Choice:"; cin>>choice;

switch(choice)

{

case 1: st[Student::getStudentCount()].getData(); break;

case 2: cout<<"\n\t All Student Database Records";

cout<<"\n \n";

cout<<"\n Record Deleted Successfully.";

}

break;

default:cout<<"\n Enter choice Inbetween 1 to 5 only.";

break;

}

cout<<"\n Do you want to continue(Y/N):"; cin.get();

cin>>ch;

}while(ch=='y'||ch=='Y');

delete []st; //dynamic memory de-allocation return 0;

}

cout<<setw(5)<<left<<"R No"<<setw(20)<<left<<"Name"<<setw(6)

<<left<<"Class"<<setw(11)<<left<<"Birth Date"<<setw(7)<<left

<<"BGroup"<<setw(25)<<left<<"Address"<<setw(11)<<"Mobile No"<<setw(4)<<left<<"S1"<<setw(4)<<left<<"S2"

<<setw(4)<<left<<"S3"<<setw(6)<<left<<"Total"<<setw(8)

<<left<<"Average"<<setw(20)<<left<<"Grade"<<endl; for(i=0;i<Student::getStudentCount();i++)

st[i].putData();

cout<<"\n \n"; break;

case 3: cout<<"Enter Roll Number to Search Record:"; cin>>key;

for(i=0,flag=0;i<Student::getStudentCount();i++)

{

if(key==st[i].getRno())

{

cout<<"\n\t Found Student Database Record"; cout<<setw(5)<<left<<"R No"<<setw(20)<<left

<<"Name"<<setw(6)<<left<<"Class"<<setw(11)

<<left<<"Birth Date"<<setw(7)<<left<<"BGroup"

<<setw(25)<<left<<"Address"<<setw(11)<<left

<<"Mobile No"<<setw(4)<<left<<"S1"<<setw(4)

<<left<<"S2"<<setw(4)<<left<<"S3"<<setw(6)<<left

<<"Total"<<setw(8) <<left<<"Average"<<setw(20) <<left<<"Grade"<<endl;

st[i].putData(); flag=1;

}

}

case 4:

break;

if(flag==0)

cout<<"\n"<<key<<" Record not present.";

cout<<"Data Entry was Done out of "<<Student::getStudentCount()

<<" / "<<cnt<<endl;

break;

case 5: cout<<"Enter Roll Number to Search Record:";

cin>>key; for(i=0,flag=0;i<Student::getStudentCount();i++)

{

if(key==st[i].getRno())

{

flag=1;

break;

}

}

else

if(flag==0)

cout<<"\n"<<key<<" Record not present.";

{

for(;i<Student::getStudentCount()-1;i++)

st[i].shift(&st[i+1]); Student::del();

Output:

Enter Student Persoanal Details:

Roll Number:1 Name:Kuldeep Class:SEA

Date of Birth (DD/MM/YY):21/08/89 Blood Group:B+

Address:Dighi-Hills, Pune Mobile Number:8668277166

Enter Student Result Data: Subject 1 Marks:67

Subject 2 Marks:78

Subject 3 Marks:87

Both S1 & S2 Object:

--

R No Name Class Birth Date BGroup Address

Mobile No S1 S2 S3 Total Average Grade

Exiting from the Class. Exiting from the Class.

Max Number of Student into Class:62 Student Database Program

1. Add New Student
2. Show all Students
3. Search Particular Student
4. How many Students Data Entry was Done?
5. Delete Student Enter your Choice:1

Enter Student Persoanal Details: Roll Number:3201 Name:Prakash

Class:SEA

Date of Birth (DD/MM/YY):21/09/99 Blood Group:AB- Address:Pune Mobile Number:8888888888

Enter Student Result Data:

Subject 1 Marks:45

Subject 2 Marks:56

Subject 3 Marks:78

Do you want to continue(Y/N):y Student Database Program

1. Add New Student
2. Show all Students
3. Search Particular Student
4. How many Students Data Entry was Done?
5. Delete Student Enter your Choice:1

Enter Student Personal Details:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Kuldeep | SEA 21/08/89 B+ | Dighi-Hills, Pune | 8668277166 67 78 87 | 232 | 77 | First with Dist |
| 1 | Kuldeep | SEA 21/08/89 B+ | Dighi-Hills, Pune | 8668277166 67 78 87 | 232 | 77 | First with Dist - |
| - |  |  |  |  |  |  |  |

R No Name Class Birth Date BGroup Address Mobile No S1 S2 S3 Total Average Grade

3202 Kailas SEA 31/01/99 B- Shikago 999999999 56 65 61 182 60 First class

Do you want to continue(Y/N):y Student Database Program

1. Add New Student
2. Show all Students
3. Search Particular Student
4. How many Students Data Entry was Done?
5. Delete Student Enter your Choice:5

Enter Roll Number to Deleting Record:3201 Record Deleted Successfully.

Do you want to continue(Y/N):y Student Database Program

1. Add New Student
2. Show all Students
3. Search Particular Student
4. How many Students Data Entry was Done?
5. Delete Student Enter your Choice:2

All Student Database Records

R NoName Class Birth Date BGroup Address Mobile No S1 S2 S3 Total Average Grade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3202 Kailas | SEA 31/01/99 B- | Shikago | 9999999999 56 65 61 182 60 | First class |
| 3203 Yogesh | SEA 23/07/99 A+ | Jaipur | 9898989898 45 44 43 132 44 | Pass class |

Do you want to continue(Y/N):y Student Database Program

1. Add New Student
2. Show all Students
3. Search Particular Student
4. How many Students Data Entry was Done?
5. Delete Student Enter your Choice:5

Enter Roll Number to Deleting Record:3205 3205 Record not present.

Do you want to continue(Y/N):y Student Database Program

1. Add New Student
2. Show all Students
3. Search Particular Student
4. How many Students Data Entry was Done?
5. Delete Student Enter your Choice:4

Data Entry was Done out of 2 / 62 Do you want to continue(Y/N):N

1. Conclusion:

From above experiment, we have studied & implemented the concept of constructor, default constructor, copy constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation operators-new and delete.

1. Review Questions & Exercises:
   1. Fill in the Blanks
      1. In C++ the **Constructor** is the construct primarily used to create objects.
      2. Members of a class object may be accessed through a pointer to the object by using the **arrow(->)** operator.
      3. A destructor has the same name as the class, but is preceded by a(n) **tilde(~)**

character.

* + 1. **this** is a special built-in pointer that is automatically passed as a hidden argument to all nonstatic member functions.
    2. When a function is declared a **friend** by a class, it becomes a member of that class.
  1. Answer the following:
     1. What are the advantages of inline function?

# *Function call overhead does not occur.*

1. ***saves the overhead of push/pop variables on the stack when the function is called.***
2. ***It also saves the overhead from the return call of a function***
   * 1. What is the difference between an instance member variable and a static member variable?

# *Instance member variables hold the values that must be referenced by more than one method, constructor or block or essential part(s) of an object state that must be present throughout the class.*

***Whereas static variable would be one copy of each class variable per class, regardless of how many objects are created from it.***

* + 1. What limitation does a static member function have?

# *It can only access data members which are static.*

* + 1. When is a copy constructor called?

# *The copy constructor is called when a new object is created from an existing object, as a copy of the existing object.*

* + 1. Assume that a class named Numbers has the following static member function declaration: static void showTotal();

Write a statement that calls the showTotal function.

# *Numbers::showTotal( ) ;*

* + 1. State the difference between delete and delete [].

# *Delete is used to release the memory occupied by an object that is no longer needed, while delete[] is used to get rid of the array’s pointer and release the memory occupied by the array.*

* 1. Programming Assignments:
     1. Circle Class

Write a Circle class that has the following member variables: radius (double) constant pi(double) initialized with the value 3.14159

The class should have the following member functions:

Default Constructor. A default constructor that sets radius to 0.0. Constructor. Accepts the radius of the circle as an argument. setRadius. A mutator function for the radius variable.

getRadius. An accessor function for the radius variable. getArea. Returns the area of the circle, which is calculated as area = pi \* radius \* radius

getDiameter. Returns the diameter of the circle, which is calculated as diameter = radius \* 2

getCircumference. Returns the circumference of the circle, which is calculated as circumference = 2 \* pi \* radius



Write a program that demonstrates the Circle class by asking the user for the circle’s radius, creating a Circle object, and then reporting the circle’s area, diameter, and circumference.

# *#include<iostream> using namespace std;*

***class Circle{ double radius; static float pi ;***

***public:***

***Circle()***

***{***

***radius = 0.0;***

***}***

***Circle(double rad)***

***{***

***radius = rad;***

***}***

***void setRadius(double rad)***

***{***

***radius = rad;***

***}***

***double getRadius()***

***{***

***return radius;***

***}***

***double getArea()***

***{***

***return (pi \* radius \* radius);***

***}***

***double getDiameter()***

***{***

***return (2 \* radius);***

***}***

***double getCircumference()***

***{***

***return (2 \* pi \* radius);***

***}***

***};***

***float Circle::pi = 3.14159; int main()***

***{***

***Circle c1; double radius;***

***cout<<"\n Enter the radius of the circle : "; cin>>radius;***

***c1.setRadius(radius);***

***cout<<"\n The Diameter of the circle is : "<<c1.getDiameter()<<endl; cout<<" The Circumference of the circle is : "<<c1.getCircumference()<<endl; cout<<" The Area of the circle is : "<<c1.getArea()<<endl;***

***return 0;***

***}***

* 1. What is wrong with this code? T \*p = new T[10]; delete p;

# *As we are allocating memory to an array of type T using the new keyword, we are supposed to use delete[ ] instead of just delete operator to release the memory occupied by the p pointer to an array of type T. Hence the code is wrong.*

1. References:
   1. E Balagurusamy Object-Oriented Programming with C++.7th edition. McGraw-

Hill Publication, ISBN 10: 9352607996 ISBN 13: 9789352607990

* 1. Tony Gaddis- “STARTING OUT WITH C++ From Control Structures through Objects”, Pearson Education, ISBN 13: 978-0-13-376939-5