**Department of Information Technology**

**Delhi Technological University**

**OBJECT ORIENTED PROGRAMMING LAB (IT-203)**



Session 2020-21(IT-3rd Semester Section-2)

**Submitted By: Submitted To:-**

**Name : Varun Kumar Mrs. Geetanjali Bhola**

**Roll No. : 2K19 / IT / 140 Assistant Professor**

**Dept. of IT, DTU**

**Group : G3**

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Program** | **Date** | **Signature** |
| 1 | To print the ASCII values of a character entered by user. | 10-8-2020 |  |
| 2 | To print alternate characters of your name. | 10-8-2020 |  |
| 3 | To create a structure student with certain attributes of your choice (read from user) and print them. | 10-8-2020 |  |
| 4 | To modify the 3 rd program to take data of 5 students and print it using array of object concept | 10-8-2020 |  |
| 5 | To modify the above program further to created a nested structure called address and print the fields from that | 10-8-2020 |  |
| 6 | Create two structures called bank\_employee and bank\_branch Take the input for all the fields | 17-8-2020 |  |
| 7 | Include the concept of nested structure | 17-8-2020 |  |
| 8 | Create a class employee with following details : name, age, empid, salary, experience,bonus | 24-8-2020 |  |
| 9 | Create a class for student with fields : name, age, marks, group\_allotted, grade,status(char) | 24-8-2020 |  |
| 10 | Create classes with the following specifications: Student\_DTU and College and calculate tax paid by the student. | 31-8-2020 |  |
| 11 | Create classes with the following specifications: Create class Car, Insurance\_policies with the following features | 31-8-2020 |  |
| 12 | Define a constructor for the class student with certain fields and  invoke the constructor in main() | 7-9-2020 |  |
| 13 | Use of Static | 14-9-2020 |  |
| 14 | WAP to show const in the different cases | 21-9-2020 |  |
| 15 | Write a C++ program to make use of object array for a class and swap the values of alternate objects | 21-9-2020 |  |
| 16 | Make an employee class; use dynamic allocation of memory to 6 objects/array of objects  and overload binary ‘+’ using both member function and friend function | 12-10-2020 |  |
| 17 | Use function overloading with class student for function Make\_project() | 12-10-2020 |  |
| 18 | Overload new and delete operators | 12-10-2020 |  |
| 19 | Overload [], =,<<,>> operators | 12-10-2020 |  |
| 20 | Implement the inheritance within classes using C++ | 26-10-2020 |  |
| 21 | Explore the order of constructor and destructor calls also see the impact of virtual keyword on constructor call with multiple inheritance | 2-11-2020 |  |
| 22 | Design a class called admission acting as abstract class with two children classes called school\_admission and college\_admission deriving from admission class. Create the virtual functions too for the same and study its impact on the given scenario | 2-11-2020 |  |
| 23 | Show the constructor destructor call with and without the virtual destructor on using the base called pointer and deleting the same | 2-11-2020 |  |
| 24 | Implement Templates and exception handling in student class while sorting marks | 9-11-2020 |  |
| 25 | Implementation of friend function and friend class | 16-11-2020 |  |

1. **To print the ASCII values of a character entered by user.**

#include <stdio.h>

int main() {

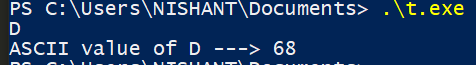
char c;

scanf("%c", &c);

printf("ASCII value of %c ---> %d", c, c);

return 0;

}



1. **To print alternate characters of your name**

#include <stdio.h>

#include<string.h>

int main() {

char myname[50];

printf("ENTER YOUR NAME : ");

fgets(myname, sizeof(myname), stdin);

int len=strlen(myname);

for(int i=0 ; i < len ; i++){

if(i%2==0) printf("%c",myname[i]);

}

return 0;

}



3. **To create a structure student with certain attributes of your choice (read from user) and print them.**

#include <stdio.h>

struct student {

char name[25];

int roll;

float marks;

};

int main() {

student s;

printf("Enter name -> ");

fgets(s.name, sizeof(s.name), stdin);

printf("Enter Roll number -> ");

scanf("%d", &s.roll);

printf("Enter marks -> ");

scanf("%f", &s.marks);

printf("Name: ");

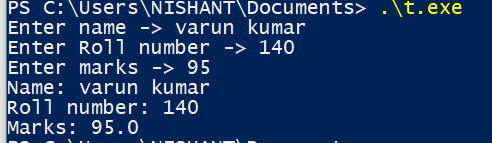
printf("%s", s.name);

printf("Roll number: %d\n", s.roll);

printf("Marks: %.1f\n", s.marks);

return 0;

}



4. **To modify the above 3 rd program to take data of 5 students and print it using array of object concept.**

#include <stdio.h>

struct student {

char name[25];

int roll;

float marks;

};

int main() {

student s[5];

for(int i=0 ;i<5 ;i++){

printf("\n%d Student :-- \n",i+1 );

printf("Enter name -> ");

gets(s[i].name);

printf("Enter Roll number -> ");

scanf("%d", &s[i].roll);

printf("Enter marks -> ");

scanf("%f", &s[i].marks);

getchar();

}

for(int i=0 ;i<5 ; i++){

printf("%d Student :-- \n",i+1 );

printf("Name: ");

printf("%s\n", s[i].name);

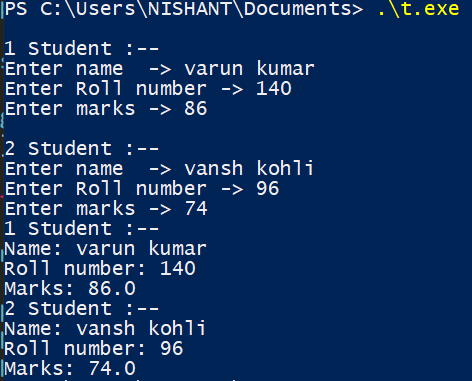
printf("Roll number: %d\n", s[i].roll);

printf("Marks: %.1f\n", s[i].marks);

}

return 0;

}



5. **To modify the above program further to created a nested structure called address and print the fields from that.**

#include <stdio.h>

struct student {

char name[25];

int roll;

float marks;

struct address{

char HouseNo[5];

char Area[50];

}a;

};

int main() {

student s[5];

for(int i=0 ;i<5 ;i++){

printf("\n%d Student :-- \n",i+1 );

printf("Enter name -> ");

gets(s[i].name);

printf("Enter Roll number -> ");

scanf("%d", &s[i].roll);

printf("Enter marks -> ");

scanf("%f", &s[i].marks);

getchar();

printf("Enter HouseNo ->");

gets(s[i].a.HouseNo);

getchar();

printf("Enter Area ->" );

gets(s[i].a.Area);

getchar();

}

for(int i=0 ;i<5 ; i++){

printf("%d Student :-- \n",i+1 );

printf("Name: ");

printf("%s\n", s[i].name);

printf("Roll number: %d\n", s[i].roll);

printf("Marks: %.1f\n", s[i].marks);

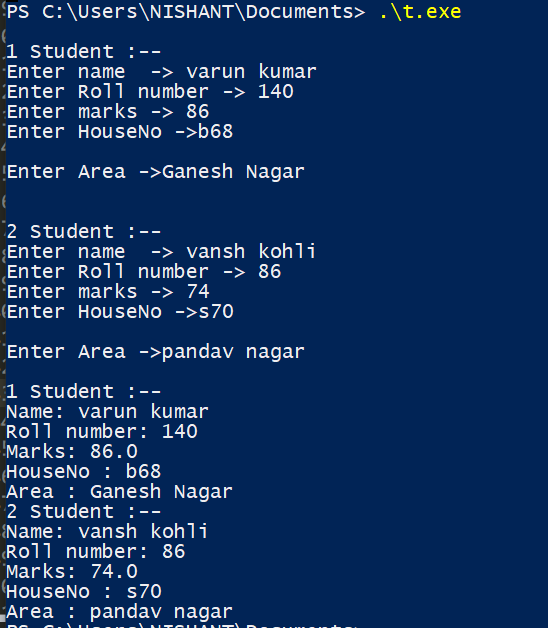
printf("HouseNo : %s\n",s[i].a.HouseNo );

printf("Area : %s\n",s[i].a.Area );

}

return 0;

}



**6. Create two structures called bank\_employee (name, empid, salary, age, city\_of\_work) and bank\_branch ( branchid, branch\_city,bonus\_offered\_to\_employees) Take the input for all the fields from user. Create a database of 5 employees and 5 branches.**

#include <stdio.h>

struct bank\_employee {

char name[25];

int empId;

int salary;

char city\_of\_work[20];

int total\_salary\_with\_bonus;

};

struct bank\_branch {

int branchId;

char branch\_city[20];

int bonus\_offered;

};

int main() {

struct bank\_employee E[5];

struct bank\_branch B[5];

for(int i=0 ;i<5 ;i++){

printf("enter name: ");

scanf("%s",&E[i].name);

printf("enter employee ID: ");

scanf("%d",&E[i].empId);

printf("enter salary: ");

scanf("%d",&E[i].salary);

printf("enter city of work: ");

scanf("%s",&E[i].city\_of\_work);

printf("enter branch ID: ");

scanf("%d",&B[i].branchId);

printf("enter branch city: ");

scanf("%s",&B[i].branch\_city);

printf("enter bonus\_offered: ");

scanf("%d",&B[i].bonus\_offered);

E[i].total\_salary\_with\_bonus = E[i].salary + B[i].bonus\_offered;

}

printf("\n");

for(int i=0 ;i<5 ; i++){

printf("Name: %s\n", E[i].name);

printf("employee ID: %d\n", E[i].empId);

printf("salary: %d\n", E[i].salary);

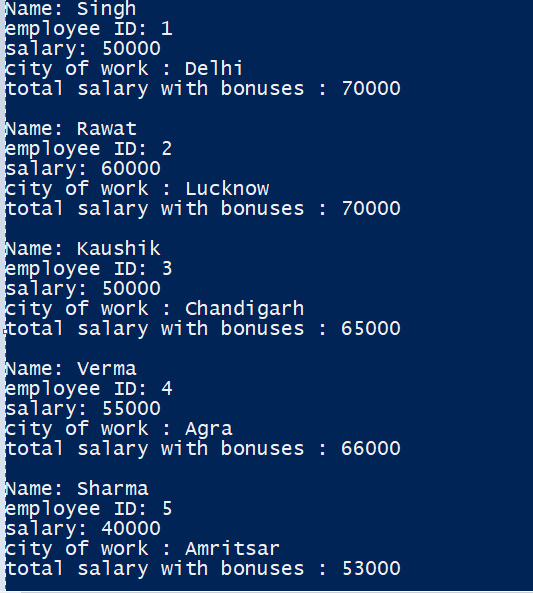
printf("city of work : %s\n",E[i].city\_of\_work);

printf("total salary with bonuses : %d\n\n",E[i].total\_salary\_with\_bonus);

}

return 0;

}



**7. Include the concept of nested structure address in it, such that each employee has an address and branch also has an address. Print the branch’s full address on the basis of employees value in city\_of\_work**

#include <stdio.h>

struct bank\_employee {

char name[25];

int empId;

int salary;

char city\_of\_work[20];

int total\_salary\_with\_bonus;

};

struct bank\_branch {

int branchId;

char branch\_city[20];

int bonus\_offered;

struct Branch\_address{

char Building\_no[5];

char Area[20];

}a;

};

int main() {

struct bank\_employee E[3];

struct bank\_branch B[3];

for(int i=0 ;i<3 ;i++){

printf("enter name: ");

scanf("%s",&E[i].name);

printf("enter employee ID: ");

scanf("%d",&E[i].empId);

printf("enter salary: ");

scanf("%d",&E[i].salary);

printf("enter city of work: ");

scanf("%s",&E[i].city\_of\_work);

printf("enter branch ID: ");

scanf("%d",&B[i].branchId);

printf("enter branch city: ");

scanf("%s",&B[i].branch\_city);

printf("enter bonus\_offered: ");

scanf("%d",&B[i].bonus\_offered);

getchar();

printf("enter branch address:-\n Enter Building\_no : " );

gets(B[i].a.Building\_no);

printf("enter Area: " );

gets(B[i].a.Area);

E[i].total\_salary\_with\_bonus = E[i].salary + B[i].bonus\_offered;

}

int employee\_id;

printf("\n");

printf("enter employee id to be searched: ");

scanf("%d",&employee\_id);

printf("\n");

for(int i=0 ;i<3 ; i++){

if(E[i].empId==employee\_id){

printf("employee name: %s\n",E[i].name);

printf("Branch\_address :-\n" );

printf("Building\_no : %s\n",B[i].a.Building\_no);

printf("Area: %s\n",B[i].a.Area );

printf("City : %s\n", B[i].branch\_city);

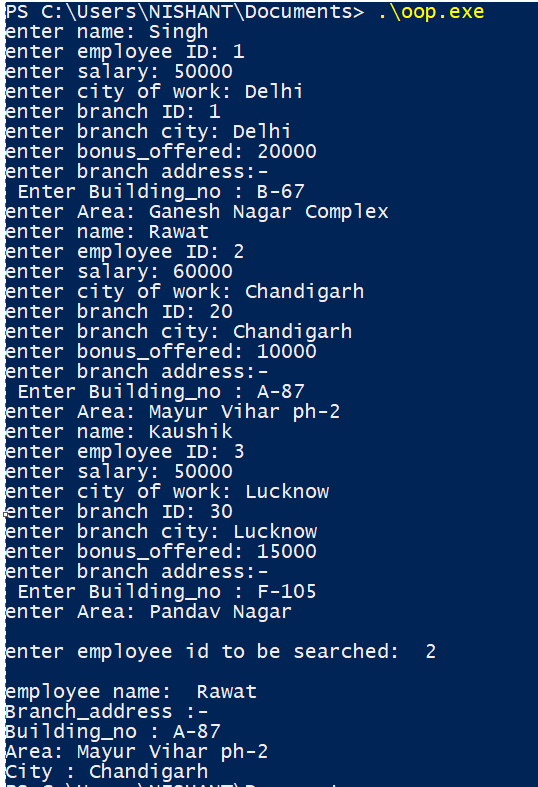
break;

}

}

return 0;

}



**8. Create a class employee with following details : name, age, empid, salary, experience,bonus**

**And use member functions : void setdata()// for setting the values //cin&gt;&gt;a;**

**Void print\_data()//cout&lt;&lt;**

**Void assign\_bonus() bonus= experience \* 2000**

**And print\_bonus() function to print the calculated value**

#include <iostream>

using namespace std;

class employee{

string name;

int age;

int empid;

int salary;

int experience;

int bonus;

public:

void setdata(){

cout<<"enter name: "; cin>>name;

cout<<"enter age: "; cin>>age;

cout<<"enter empid: "; cin>>empid;

cout<<"enter salary: "; cin>>salary;

cout<<"enter experience: "; cin>>experience;

cout<<endl;

}

void print\_data(){

cout<<"name: "<<name<<endl;

cout<<"age: "<<age<<endl;

cout<<"empid: "<<empid<<endl;

cout<<"salary: "<<salary<<endl;

cout<<"experience: "<<experience<<endl;

}

void assign\_bonus(){

bonus=experience\*2000;

}

void print\_bonus(){

cout<<"bonus: "<<bonus<<endl;

}

};

int main(){

employee E[2];

for(int i=0 ;i<2;i++){

E[i].setdata();

E[i].assign\_bonus();

}

for(int i=0;i<2;i++){

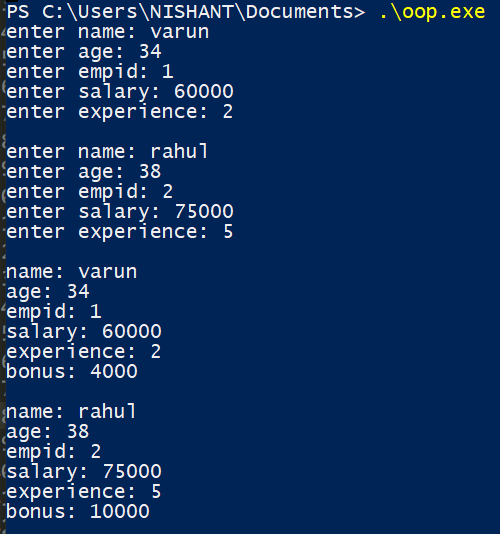
E[i].print\_data();

E[i].print\_bonus();

}

return 0;

}



**9. Create a class for student with fields : name, age, marks, group\_allotted, grade,status(char)**

**Take the values from user**

**On the basis of first character of his name, he should be allotted a group: if(a-m): group A, B**

**otherwise**

**On the basis of marks allot grade: 90&gt; grade 0; 80-89- A; 70-79-B and so on..**

**On the basis of age, his status should be considered: E/N ie eligible to vote or not**

#include <iostream>

using namespace std;

class student{

string name;

int age;

float marks;

char group\_allotted;

char grade;

char status;

public:

void setdata(){

cout<<"enter name: "; cin>>name;

cout<<"enter age: "; cin>>age;

cout<<"enter marks: "; cin>>marks;

}

void allot\_group(){

if(name[0] >= 'a' and name[0] <= 'm' )

group\_allotted='A';

else group\_allotted='B';

}

void allot\_grade(){

if(marks >= 90) grade='O';

else if(marks >= 80) grade='A';

else if(marks >=70) grade='B';

else if(marks >= 60) grade='C';

else if(marks >= 50) grade='D';

else if(marks >= 40) grade='E';

else grade='F';

}

void allot\_status(){

if(age >= 18) status='E';

else status='N';

}

void print\_data(){

cout<<"name: "<<name<<endl;

cout<<"group\_allotted: "<<group\_allotted<<endl;

cout<<"status: "<<status<<endl;

cout<<"grade: "<<grade<<endl;

}

};

int main(){

student S[2];

for(int i=0;i<2;i++){

S[i].setdata();

S[i].allot\_status();

S[i].allot\_grade();

S[i].allot\_group();

cout<<endl;

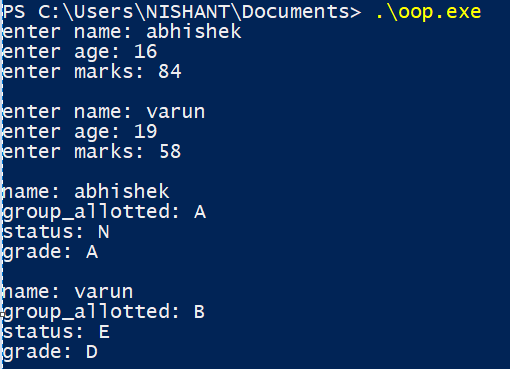
}

for(int i=0;i<2;i++){

S[i].print\_data();

cout<<endl;

}



return 0;

}

**10. Create classes with the following specifications: Student\_DTU and College and calculate tax paid by the student.**

#include <iostream>

using namespace std;

class student\_DTU{

string name;

int roll,marks,Package\_offered,Age;

char P\_grade,grade;

int College\_id;

public:

void Package\_grade(){

if(Package\_offered>15) P\_grade='A';

else P\_grade='B';

}

void Name\_ascii(){

cout<<(int)name[0];

}

void Marks\_grade(){

if(marks>80) grade='A';

else if(marks>60) grade='B';

else grade='C';

}

student\_DTU(){

roll=0; marks=0;

Package\_offered=0;

Age=0;

cout<<"Welcome - ";

}

void get\_data(){

cout<<"Enter name , roll , marks , Package\_offered , Age , College\_id:- "<<endl;

cin>>name>>roll>>marks>>Package\_offered>>Age>>College\_id;

}

int Package\_offered\_get(){

return Package\_offered;

}

int Age\_get(){

return Age;

}

} S ; //global object

class college{

string name;

int College\_id;

string State;

int Sem\_fees\_btech;

int Percentage\_stud\_placement;

public:

void Remarks\_for\_college\_print(){

if(Percentage\_stud\_placement>80) cout<<"Good";

else cout<<"not good";

}

college(){

College\_id=0; Sem\_fees\_btech=0;

Percentage\_stud\_placement=0;

}

void get\_data(){

cout<<"Enter name , College\_id , State , Sem\_fees\_btech , Percentage\_stud\_placement:- "<<endl;

cin>>name>>College\_id>>State>>Sem\_fees\_btech>>Percentage\_stud\_placement;

}

int Sem\_fees\_btech\_get(){

return Sem\_fees\_btech;

}

} C; // global declare

int calculate\_Tax(){

int tax=abs(((S.Package\_offered\_get())/S.Age\_get()) - (C.Sem\_fees\_btech\_get()\*8));

return tax;

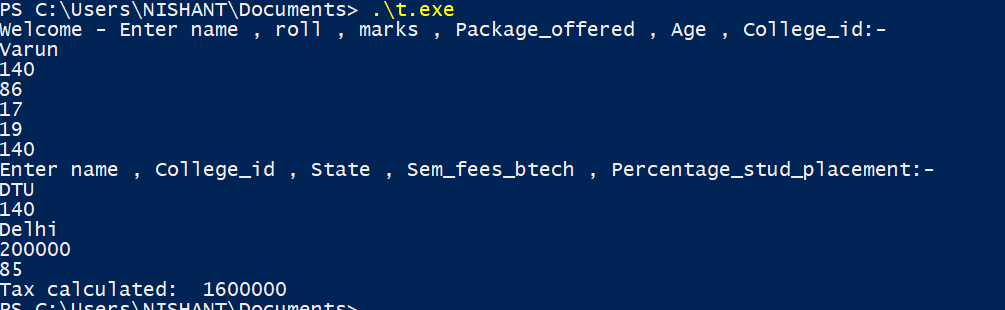
}

int main(){

S.get\_data();

C.get\_data();

cout<<"Tax calculated: "<<calculate\_Tax();

return 0; }

**11. Create classes with the following specifications: Create class Car, Insurance\_policies with the following features**

#include <iostream>

using namespace std;

class car{

int model\_no;

string name;

int price;

float insurance\_expected;

public:

float cal\_insurance(){

insurance\_expected=(float)price\*0.010;

return insurance\_expected;

}

void get\_data(){

cout<<"enter model\_no , name and price :- "<<endl;

cin>>model\_no>>name>>price;

}

void print(){

cout<<endl;

cout<<"model\_no: "<<model\_no<<endl;

cout<<"name: "<<name<<endl;

cout<<"price: "<<price<<endl;

cout<<"insurance\_expected: "<<insurance\_expected<<endl;

}

car(){

model\_no=0; price=0;

insurance\_expected=0;

}

} C ; //global object

class insurance\_policy{

int policy\_no;

string name\_of\_policy;

int amount;

float discount;

public:

float cal\_discount(){

discount=(float)amount\*0.10;

return discount;

}

void get\_data(){

cout<<"enter policy\_no , name\_of\_policy , amount : - "<<endl;

cin>>policy\_no>>name\_of\_policy>>amount;

}

void print(){

cout<<endl;

cout<<"policy\_no: "<<policy\_no<<endl;

cout<<"name\_of\_policy: "<<name\_of\_policy<<endl;

cout<<"amount: "<<amount<<endl;

cout<<"discount: "<<discount<<endl;

}

insurance\_policy(){

policy\_no=0;

amount=0;

discount=0;

}

} P ; //global object

class service\_station{

string name,location;

float average\_bill;

float discount;

public:

float cal\_discount(){

discount=average\_bill\*0.10;

return discount;

}

void get\_data(){

cout<<"enter name , location , average\_bill:- "<<endl;

cin>>name>>location>>average\_bill;

}

void print(){

cout<<endl;

cout<<"name: "<<name<<endl;

cout<<"location: "<<location<<endl;

cout<<"average\_bill: "<<average\_bill<<endl;

cout<<"discount: "<<discount<<endl;

}

} S ; //global object

class person\_data{

float my\_car\_insurance;

float my\_insur\_discount;

float my\_service\_station\_amount;

float total\_expenditure;

public:

void car\_total\_Expenditure(){

total\_expenditure=my\_car\_insurance + my\_insur\_discount + my\_service\_station\_amount;

}

void set\_data(){

my\_car\_insurance=C.cal\_insurance();

my\_insur\_discount=P.cal\_discount();

my\_service\_station\_amount=S.cal\_discount();

}

void print(){

cout<<endl;

cout<<"my\_car\_insurance: "<<my\_car\_insurance<<endl;

cout<<"my\_insur\_discount: "<<my\_insur\_discount<<endl;

cout<<"my\_service\_station\_amount: "<<my\_service\_station\_amount<<endl;

cout<<"total\_expenditure: "<<total\_expenditure<<endl;

}

} PD ; //global object

int main(){

C.get\_data();

P.get\_data();

S.get\_data();

C.cal\_insurance();

P.cal\_discount();

S.cal\_discount();

PD.set\_data();

PD.car\_total\_Expenditure();

C.print();

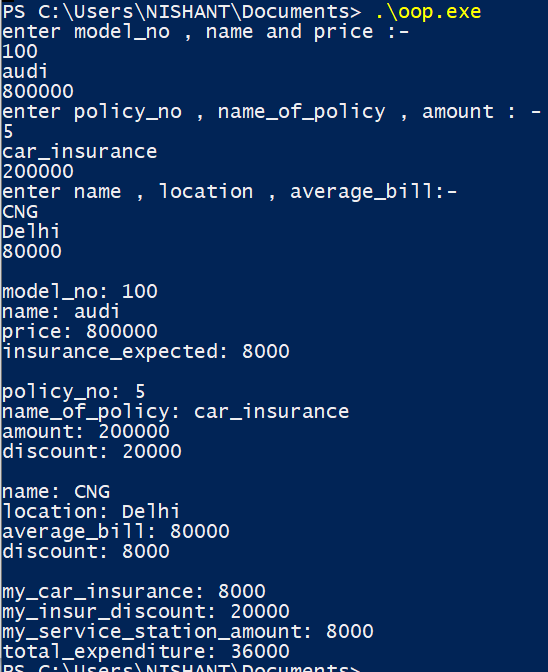
P.print();

S.print();

PD.print();

return 0;

}



**12 . Define a constructor for the class student with certain fields and**

**invoke the constructor in main()**

#include <iostream>

using namespace std;

class student{

string name;

int roll\_no;

float marks;

student(){

cout<<" default private constructor called"<<endl<<endl;

}

public:

int get\_roll\_no(){

return roll\_no;

}

float get\_marks(){

return marks;

}

string get\_name(){

return name;

}

student(string n){

cout<<"parameterized with one parameter called "<<endl<<endl;

name=n;

roll\_no=0; marks=0;

}

student(int r){

cout<<"parameterized with one parameter called "<<endl<<endl;

roll\_no=r;

}

student(float m){

cout<<"parameterized with one parameter called"<<endl<<endl;

marks=m;

}

student(string n, int r , float m){

cout<<"parameterized with three parameters called"<<endl<<endl;

student s;

name=n;

roll\_no=r;

marks=m;

}

student(const student &s){

cout<<"copy constructor called "<<endl<<endl;

name=s.name;

roll\_no=s.roll\_no;

marks=s.marks;

}

~student(){

cout<<" DESTRUCTOR called and object destroyed"<<endl<<endl;

}

};

int main(){

student s1("varun" , 140 , 94 );

student s2("vansh");

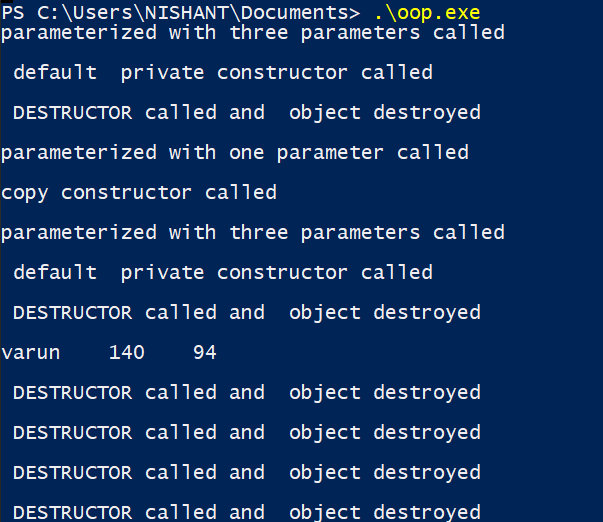
student s3(s1) ;

student s4("rahul" , 125 , 74);

cout<<s3.get\_name()<<" "<<s3.get\_roll\_no()<<" "<<s3.get\_marks()<<endl<<endl;

return 0;

}



**13. Use of static**

#include <iostream>

using namespace std;

class employee{

int empid;

string name;

int salary;

static int bonus;

employee(string n , int e ,int s){

cout<<"private constructer called "<<endl;

name=n;

empid=e;

salary=s;

}

public:

static int no\_of\_objects;

void get\_data(){

cout<<"Enter name , empid , salary: "<<endl;

cin>>name>>empid>>salary;

}

void print\_data(){

cout<<"name: "<<name<<endl;

cout<<"empid: "<<empid<<endl;

cout<<"salary: "<<salary<<endl;

}

int bonus\_offered\_non\_static(){ //non static function accessing static variable

bonus=(salary\*2)/10000;

return bonus; //WORKS

}

// static int bonus\_offered\_static(){ // static accessing non static variable

// cout<<"static member function called "<<endl;

// bonus=(salary\*2)/10000;

// return bonus; //ERROR

// }

static int return\_bonus(){ // static accessing static variable

cout<<"static member function called "<<endl;

return bonus; // WORKS

}

static void private\_Call(string n,int e ,int s){

employee(n,e,s); // static member function calling private constructor

}

employee(){

cout<<"default constructor called"<<endl;

no\_of\_objects++;

}

};

int employee::no\_of\_objects=0;

int employee::bonus=1000;

int main(){

employee e1,e2,e3;

e1.get\_data();

cout<<"objects created: "<<employee::no\_of\_objects<<endl;

cout<<"bonus\_offered: "<<employee::return\_bonus()<<endl; // static member function called

static employee e4; // static object created

// It's constructed until the end of the program

e4.get\_data();

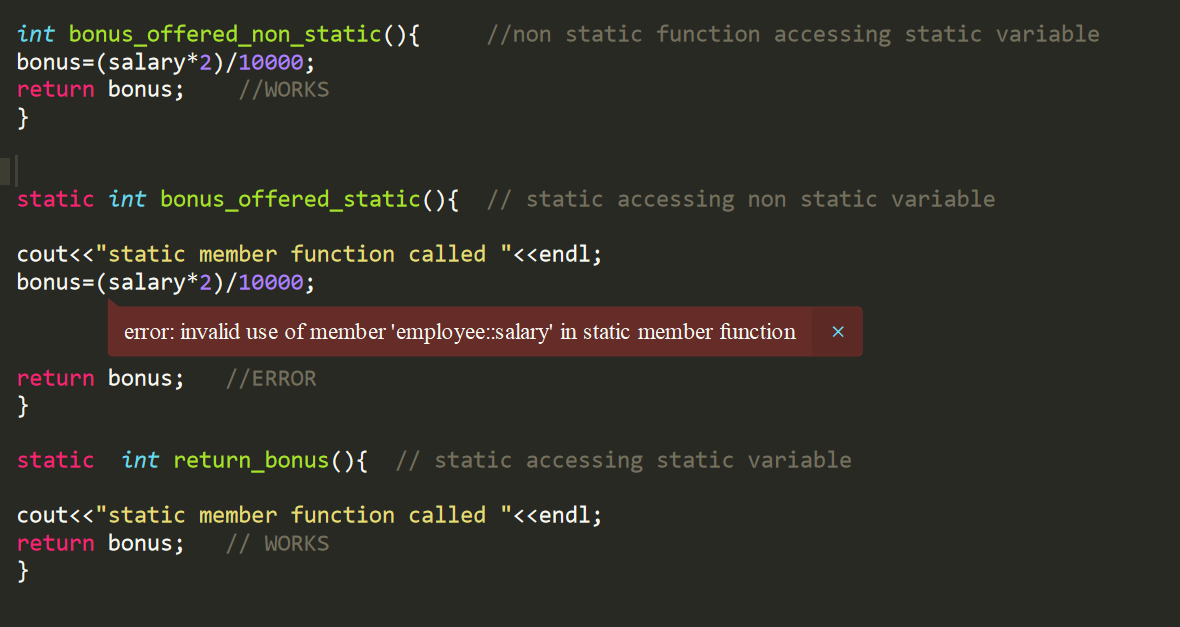
employee e5;

e5.private\_Call("vansh" , 6 , 100000); // static funbction

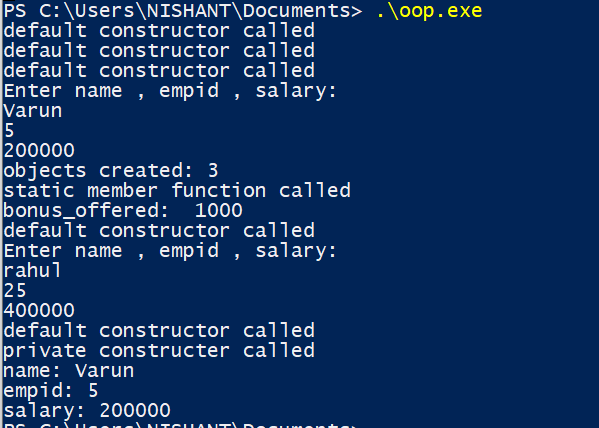
e1.print\_data();

return 0;

}



**ERROR WHILE STATIC MEMBER FUNCTION ACCESSING NON STATIC VARIABLES**



**14. WAP to show const in the following cases:**

**Const variable, function, object, const pointer, pointer to const**

#include <iostream>

using namespace std;

class student{

const string college; //constant variable

string name;

public:

student(string n):college("DELHI TECHNOLOGICAL UNIVERSITY")

{

cout<<"DECLARED CONSTANT variable for "<<n<<" \n";

name=n;

}

void print\_college() const //constant function

{

cout<<college<<endl;

}

};

int main()

{

const student s1("STUDENT 1"); //constant object

s1.print\_college();

student s2("STUDENT 2");

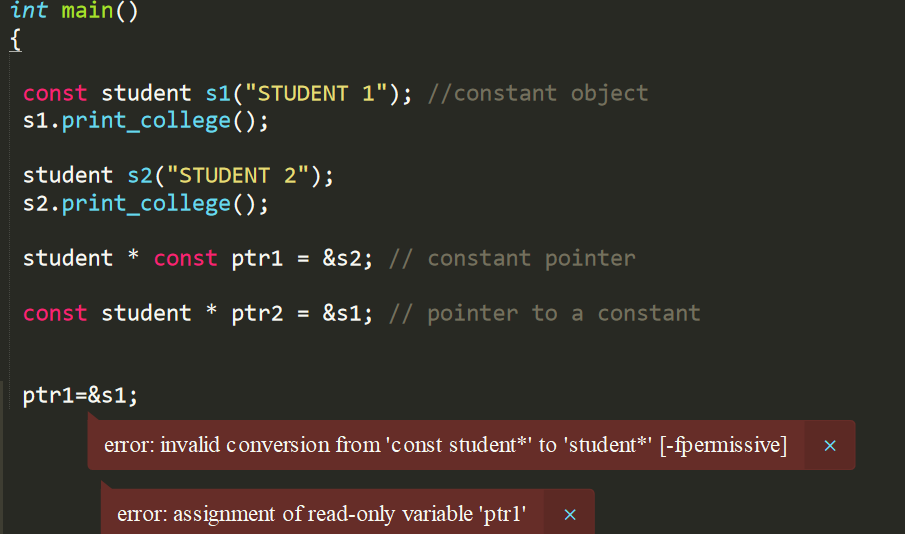
s2.print\_college();

student \* const ptr1 = &s2; // constant pointer

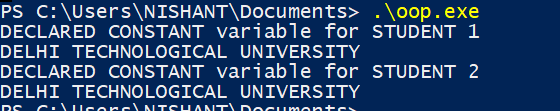
const student \* ptr2 = &s1; // pointer to a constant

// ptr1=&s1;

return 0; }



**ERROR WHILE CONSTANT POINTER MODIFIES**



**15. . Write a c++ program to make use of object array for a class and swap the values of alternate**

**objects.**

**Eg: take 10/even number of objects in array.**

**Get/set their values using constructor(s)**

**Create a swap function taking objects as arguments swap(s[0],s[1]), swap(s[2],s[3])… so on**

**After that print their swapped values**

**Use const/static to support the code.**

**NOTE: try to make the code generic.**

#include <iostream>

using namespace std;

class student{

int roll\_no;

string name;

public :

student(int d=-1,string n="UNKNOWN")

{

roll\_no=d;

name=n;

}

void print() const

{

cout<<"roll\_no:\t"<<roll\_no<<endl;

cout<<"NAME:\t"<<name<<endl;

}

};

void swap(student &s1,student &s2)

{

student temp=s1;

s1=s2;

s2=temp;

}

int main()

{

int size=5;

student a[size]={ student(140,"varun"),student(122,"VANSH"),student(100,"RAHUL"),student(125,"PIYUSH"),student(151,"VEDANT")};

for(int i=0;i<size;i+=2)

{

if(i+1<size)

swap(a[i],a[i+1]);

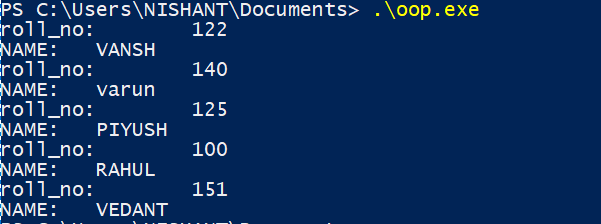
}

for(int i=0;i<size;i++)

a[i].print();

return 0;

}



**16. Make an employee class; use dynamic allocation of memory to 6 objects/array of objects and**

**calculate the average of salaries of employees on same designation, assuming that there are 3**

**distinct designations and there are two employees serving on same designation. Overload binary ‘+’**

**using both member function and friend function.**

**Ie. (E1+E2)/2; use this expression to calculate average of salaries for a same designation.**

#include<iostream>

#include<string>

using namespace std;

class Employee

{

string emp\_number;

int salary;

string designation;

public:

Employee()

{

int t;

cout<<"Enter the employee number : ";

cin>>emp\_number;

cout<<"Enter the salary : ";

cin>>salary;

cout<<"choose your designation : 1.Asst manager || 2.clerk || 3.vp"<<endl;

cin>>t;

switch (t)

{

case 1: designation="Asst manager";

break;

case 2: designation ="clerk";

break;

case 3: designation ="vp";

break;

default: designation ="none";

break;

}

}

int get\_salary()

{ return this->salary; }

string get\_designation()

{ return this->designation; }

int operator + (Employee const &e1) // operator overloading

{

return (this->salary + e1.salary);

}

};

void average\_salary(Employee e1, Employee e2) // use of operator overloading

{

cout<<"\nAvg salary for "<<e1.get\_designation()<<" is "<<(e1+e2)/2;

}

int main()

{

Employee \* E = new Employee[6];

for (int i = 0; i < 6; i++)

{

E[i];

}

for (int i = 0; i < 5; i=i+2)

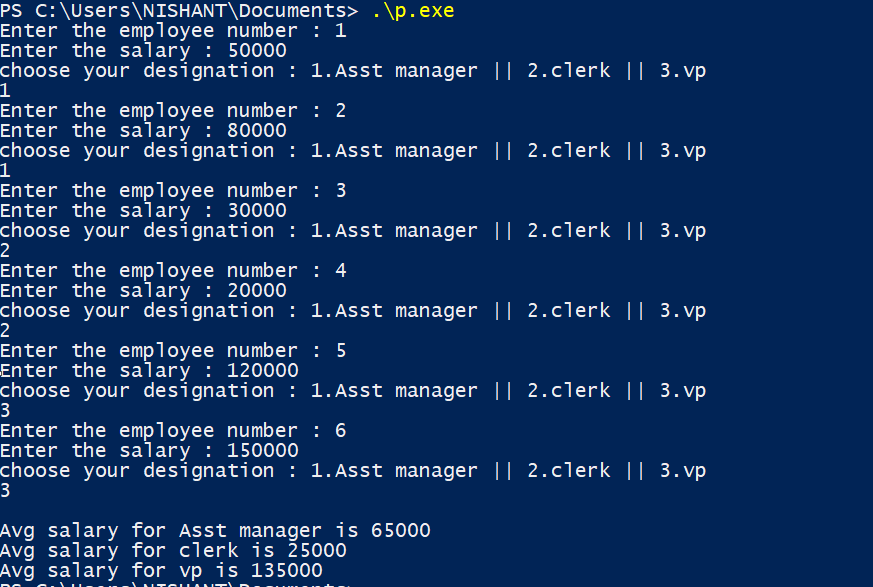
{

average\_salary(E[i],E[i+1]); // displaying the avg salary

}

return 0;

}



17.  **Use function overloading with class student for function Make\_project().**

**Make\_project(): for 1st year students**

**Make\_project(one argument): 2nd year student and so on...... till final year students.**

#include<iostream>

#include<string>

using namespace std;

class Student

{

string name;

public:

Student()

: name{"deafult name"}

{}

Student(string name\_val)

: name{name\_val}

{}

void make\_project()

{

cout<<"\nThis 1st year project is done by "<<this->name<<endl;

}

void make\_project(float year1) // cg for first year

{

cout<<"\nThis 2nd year project is done by "<<this->name<<endl;

}

void make\_project(float year1, float year2) // cg for 1st and 2nd year

{

cout<<"\nThis 3rd year project is done by "<<this->name<<endl;

}

void make\_project(float year1, float year2, float year3) // cg for all the 3 years

{

cout<<"\nThis 4th year project is done by "<<this->name<<endl;

}

};

int main()

{

Student s1("VARUN");

Student s2("VANSH");

Student s3("JAI");

Student s4("KUNAL");

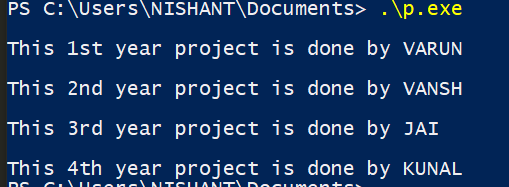
s1.make\_project();

s2.make\_project(9.5);

s3.make\_project(8.6,8.0);

s4.make\_project(7.8,9.8,9.2);

}



**18 .Overload new and delete operators and discuss the role of global new and global delete while**

**overloading it.**

#include<iostream>

using namespace std;

class employee

{

string name;

int age;

public:

employee()

{

cout<< "Constructor is called\n\n" ;

}

employee(string name, int age)

{

this->name = name;

this->age = age;

}

void display()

{

cout<< "Name:" << name << endl << endl;

cout<< "Age:" << age << endl << endl;

}

void \* operator new(size\_t size)

{

cout<< "Overloading new operator with size: " << size << endl << endl;

void \* p = ::new employee();

return p;

}

void operator delete(void \* p)

{

cout<< "Overloading delete operator " << endl << endl;

free(p);

}

};

int main()

{

employee \* e1 = new employee("VANSH", 24);

employee \* e2 = new employee("RAHUL",25);

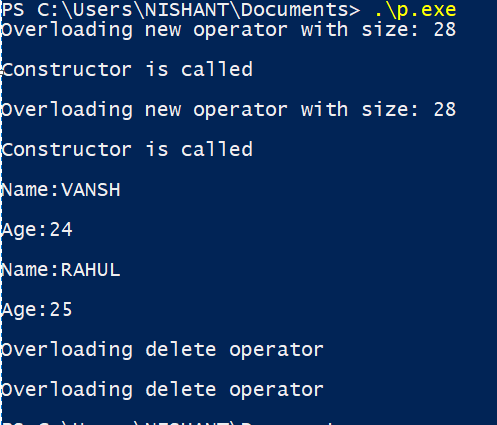
e1->display();

e2->display();

delete e1;

delete e2;

}



**19. Overload =,<<,>> operators**

#include <iostream>

using namespace std;

class Distance {

private:

int feet;

int inches;

public:

Distance() {

feet = 0;

inches = 0;

}

Distance(int f, int i) {

feet = f;

inches = i;

}

friend ostream &operator<<( ostream &output, const Distance &D ) {

output << "Feet : " << D.feet << " Inches : " << D.inches;

return output;

}

friend istream &operator>>( istream &input, Distance &D ) {

input >> D.feet >> D.inches;

return input;

}

void operator = (const Distance &D ) {

feet = D.feet;

inches = D.inches;

}

};

int main() {

Distance D1(15, 5), D2(8, 22), D3;

cout << "Enter the value of object : ";

cin >> D3;

cout << "First Distance: " << D1 << endl;

cout << "Second Distance: " << D2 << endl;

cout << "Third Distance: " << D3 << endl;

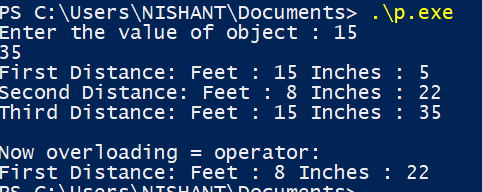
cout<<" \nNow overloading = operator: "<< endl;

D1 = D2;

cout << "First Distance: "<< D1 << endl;

return 0;

}



**OVERLOADING [] OPERATOR**

#include <iostream>

using namespace std;

const int SIZE = 10;

class safearay {

private:

int arr[SIZE];

public:

safearay() {

register int i;

for(i = 0; i < SIZE; i++) {

arr[i] = i;

}

}

int &operator[](int i) {

if( i > SIZE ) {

cout << "Index out of bounds" <<endl;

return arr[0];

}

return arr[i];

}

};

int main() {

safearay A;

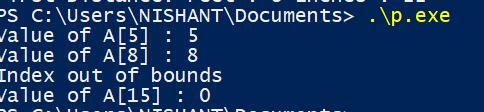
cout << "Value of A[5] : " << A[5] <<endl;

cout << "Value of A[8] : " << A[8]<<endl;

cout << "Value of A[15] : " << A[15]<<endl;

return 0;

}



**20. Implement the following using inheritance and its types in C++. Choose the data . members and**

**member functions as per your choice. Also explore the following points (among the classes of your**

**choice):**

**1) What is protected members of a class?**

**2) Different access modifiers (private, public, protected)**

**3) Are friend functions inherited?**

**4) Are constructors, destructors inherited?**

#include<iostream>

using namespace std;

class person{

int height;

int weight;

public:

person(){

cout<<"person class constructor called!!!! \n\n";

height=0;

weight=0;

}

};

class student : protected person{

protected:

string name;

public:

student(string n){

cout<<"student class constructor called!!!!\n\n";

name = n;

}

};

class collge\_student : public student{

int roll\_no;

public:

collge\_student(string n,int r) : student(n){

cout<<"collge\_student class constructor called!!!\n\n";

roll\_no=r;

}

};

class employee : public person{

protected:

string name;

int emp\_id;

public:

employee(string n , int e){

cout<<"employee class constructor called!!! \n\n";

name = n;

emp\_id=e;

}

};

class college\_employee : public employee{

int salary;

public:

college\_employee(string n,int e,int s) : employee(n,e){

cout<<"college\_employee class constructor called!!! \n\n";

salary =s;

}

friend int bonus(int salary);

};

class college\_people: public collge\_student , protected college\_employee{

string college\_name;

public:

college\_people(string na,int r,string n,int e,int s ,string c): collge\_student(na,r),college\_employee(n,e,s) {

cout<<"college\_people class constructor called!!! \n\n";

college\_name=c;

}

};

int bonus(int salary){

int b=(salary\*2)/100;

return b;

}

int main(){

college\_people C("varun",124,"piyush",450,50000,"DTU");

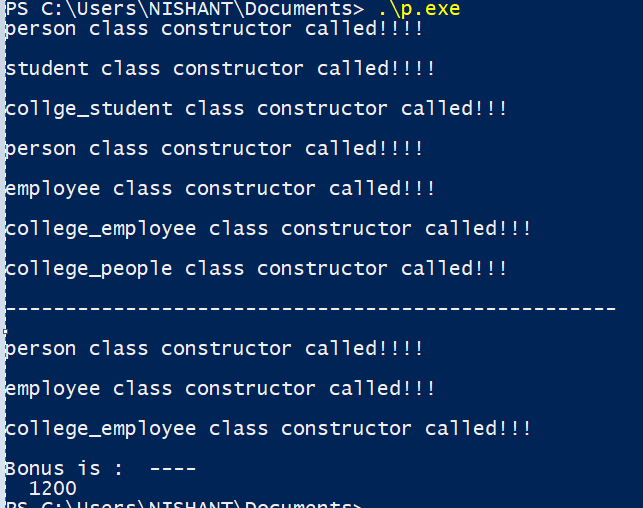
cout<<"---------------------------------------------------\n\n";

college\_employee E("rahul",694,60000);

cout<<"Bonus is : ---- \n "<<bonus(60000);

return 0;

}



**21. Explore the order of constructor and destructor calls also see the impact of virtual keyword on constructor call with multiple inheritance**

#include <iostream>

using namespace std;

class Animal {

public:

int A;

Animal()

{

cout<<"Animal class constructor called !!!\n\n" ;

}

~Animal(){

cout<<"Animal class destructor called !!!\n\n" ;

}

};

class Herbivores : virtual public Animal {

public :

int H;

Herbivores(){

cout<<"Herbivores class constructor called !!!\n\n" ;

}

~Herbivores(){

cout<<"Herbivores class destructor called !!!\n\n" ;

}

};

class Carnivores : virtual public Animal {

public:

int C;

Carnivores(){

cout<<"Carnivores class constructor called !!!\n\n" ;

}

~Carnivores(){

cout<<"Carnivores class destructor called !!!\n\n" ;

}

};

class Omnivores : public Herbivores, public Carnivores {

public:

int O;

Omnivores(){

cout<<"Omnivores class constructor called !!!\n\n" ;

}

~Omnivores(){

cout<<"Omnivores class destructor called !!!\n\n" ;

}

};

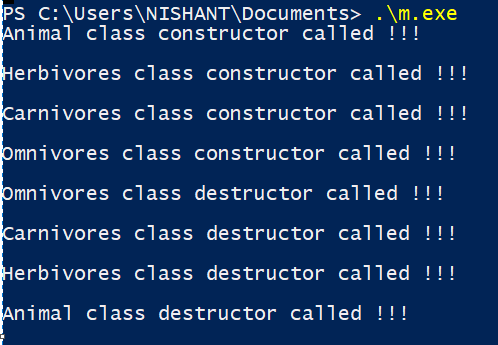
int main()

{

Omnivores obj;

return 0;

}



**22. Design a class called admission acting as abstract class with two children classes called school\_admission and college\_admission deriving from admission class. Create the virtual functions too for the same and study its impact on the given scenario**

#include <iostream>

#include<string>

using namespace std;

class admission

{

public:

string name;

virtual void procedure\_details()=0;

virtual void doc\_req()=0;

void virtual contact()=0;

};

class college\_admission:public admission

{

public:

college\_admission()

{

name= "Delhi Technological University, Delhi\n";

}

void procedure\_details()

{

cout<<"\t\t\t\tSteps to follow\n";

cout<<"Step 1:\nFee Payment\n\n";

cout<<"Step 2:\nRegistration\n\n";

cout<<"Step 3:\nLogin to JAC Delhi Account\n\n";

cout<<"Step 4:\nChoice Filling\n\n";

cout<<"Step 5:\nChoice Locking and Printing of Locked Choices\n\n";

cout<<"Step 6:\nReporting at Allotted Institute\n\n\n";

}

void doc\_req()

{

cout<<"\t\t\t\tList of Documents Requried\n";

cout<<"1.Applicable fee payment receipt of Rs. 10,000\n";

cout<<"2.Three passport size photographse\n";

cout<<"3.Self-attested copies of Admit Card and Score Card of JEE Main 2020\n";

cout<<"4.Original marksheet of the qualifying examination i.e. Class 12th\n";

cout<<"5.Original Date of Birth certificate as indicated in High School examination i.e. Class 10th\n";

cout<<"6.Photo Identity proof of both the parents\n";

cout<<"7.Photocopy of Residential Address Proof\n\n\n";

}

void contact()

{

cout<<"For any query kindly contact:\n";

cout<<"Fax: +91-11-27871023\nPhone: +91-11-27871018\n\n\n";

}

};

class school\_admission:public admission

{

public:

school\_admission()

{

name="Delhi Public School Surat\n";

}

void procedure\_details()

{

cout<<"\t\t\t\tSteps to follow\n";

cout<<"Step 1:\nCOMPLETE ONLINE REGISTRATIN FORM\nwww.dpssurat.net\n\n";

cout<<"Step 2:\nKNOW EACH OTHER SESSION\nInteraction with school committee\n\n";

cout<<"Step 3:\nADMISSION STATUS\nProvisional Selection Status\n\n";

cout<<"Step 4:\nCOMPLETE ADMISSION FORMALITIES\nAs per the schdule\n\n";

}

void doc\_req()

{

cout<<"\t\t\t\tList of Documents Requried\n";

cout<<"1.Completed registration form\n";

cout<<"2.Copy of birth certificate\n";

cout<<"3.Transfer certificate\n";

cout<<"4.Mark sheet of the last exam\n";

cout<<"5.Passport size photograph\n";

cout<<"6.Photo Identity proof of both the parents\n";

cout<<"7.Photocopy of Residential Address Proof\n\n\n";

}

void contact()

{

cout<<"For any query kindly contact:\n";

cout<<"Email:info@dpssurat.net\n7600057383 / 91-261-2654014 / 265402\n\n\n";

}

};

int main() {

college\_admission \*c1=new college\_admission;

school\_admission \*s1=new school\_admission;

int x;

do

{

cout<<"1.School Admission\t\t2.College Admission\t\t3.Exit\n";

cout<<"\t\t\t\tEnter your choice:";

cin >>x;

switch(x)

{

case 1:

{

int a;

cout<<"\n\n\t\t\t\t"<<s1->name<<endl<<endl;;

do

{

cout<<"1.Procedure Details\t2.Documents Requried\t3.Contact\t4.Exit\n";

cout<<"\t\t\t\tEnter your choice:";

cin >>a;

switch(a)

{

case 1:

{

s1->procedure\_details();

break;

}

case 2:

{

s1->doc\_req();

break;

}

case 3:

{

s1->contact();

break;

}

case 4:

{

cout<<"Thank You for visiting "<<s1->name<<endl;

break;

}

default:

{

cout<<"Please, choose correct option!\n";

}

}

}

while(a!=4);

}

case 2:

{

int a;

cout<<"\n\n\t\t\t\t"<<c1->name<<endl<<endl;

do

{

cout<<"1.Procedure Details\t2.Documents Requried\t3.Contact\t4.Exit\n";

cout<<"\t\t\t\tEnter your choice:";

cin >>a;

switch(a)

{

case 1:

{

c1->procedure\_details();

break;

}

case 2:

{

c1->doc\_req();

break;

}

case 3:

{

c1->contact();

break;

}

case 4:

{

cout<<"Thank You for visiting "<<c1->name<<endl;

break;

}

default:

{

cout<<"Please, choose correct option!\n";

}

}

}

while(a!=4);

}

case 3:

{

cout<<"Thanks for visiting\n";

break;

}

default:

{

cout<<"\n";

}

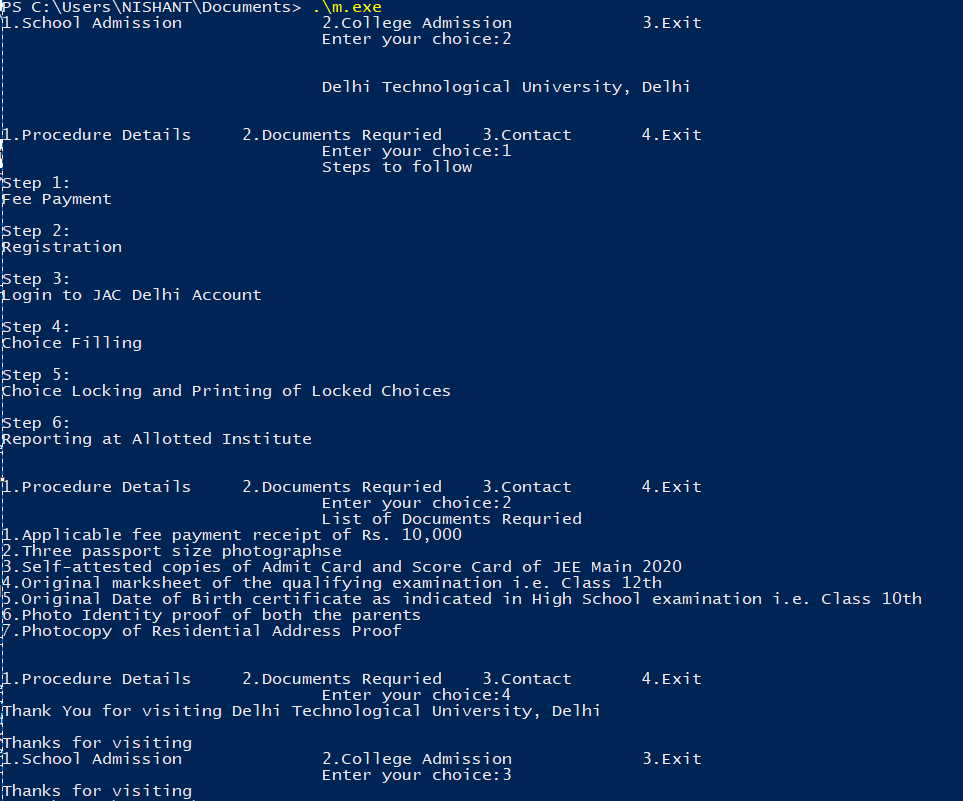
}

}

while(x!=3);

return 0;

}



**23. Show the constructor destructor call with and without the virtual destructor on using the base called pointer and deleting the same**

#include<iostream>

using namespace std;

class A

{

public:

A()

{

cout<<"Constructor of A\n";

}

virtual ~A()

{

cout<<"Destructor of A\n";

}

};

class B:public A

{

public:

B()

{

cout<<"Constructor of B\n";

}

~B()

{

cout<<"Destructor of B\n";

}

};

class C

{

public:

C()

{

cout<<"Constructor of C\n";

}

~C()

{

cout<<"Destructor of C\n";

}

};

class D:public C

{

public:

D()

{

cout<<"Constructor of D\n";

}

~D()

{

cout<<"Destructor of D\n";

}

};

int main()

{

cout<<"\t\t\tWITH VIRTUAL DESTRUCTOR\n";

A \*ob1=new B;

cout<<endl;

delete ob1;

cout<<endl;

cout<<"\t\t\tWITHOUT VIRTUAL DESTRUCTOR\n";

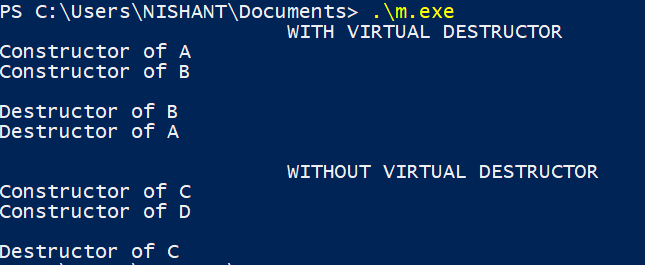
C\* ob2=new D;

cout<<endl;

delete ob2;

return 0;

}



**24. Create a student class which can accept marks as int/float/grades.**

**Student**

**----marks**

**----roll number**

**Perform a sort function based on their marks. Implement exception handling within.**

#include<iostream>

using namespace std;

template <class T>

class student

{

T marks[5];

int rollno;

string name;

public:

student(int r,string s)

{

rollno = r;

name = s;

cout<<"Name : "<<name<<endl;

cout<<"Roll no : "<<rollno<<endl;

}

void get\_marks()

{

for(int i = 0; i < 5; i++)

{

cin>>marks[i];

}

try

{

for(int i = 0;i< 5; i++)

{

if(marks[i]<0)

{

throw marks[i];

}

}

sort\_marks();

print\_marks();

}

catch(T m)

{

cout<<"Marks cannot be negative "<<m<<endl;

}

}

void sort\_marks()

{

for (int i = 0; i < 5; i++)

{

for (int j = i+1; j < 5; j++)

{

if (marks[i] > marks[j])

{

T temp;

temp = marks[i];

marks[i] = marks[j];

marks[j] = temp;

}

}

}

}

void print\_marks()

{

cout<<"sorted marks = ";

for (int i = 0; i < 5; i++)

{

cout<<marks[i]<<" ";

}

cout<<endl;

}

};

int main(){

student<int> I(49,"VARUN");

cout<<"Enter integer marks: ";

I.get\_marks();

cout<<endl;

student<float> F(35,"RAHUL");

cout<<"Enter floating marks: ";

F.get\_marks();

cout<<endl;

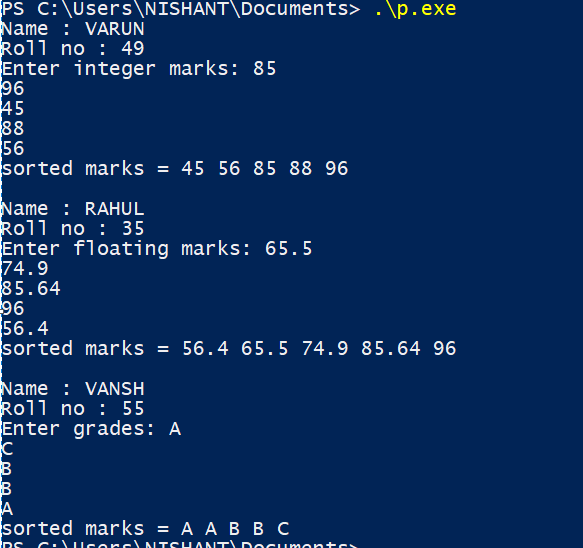
student<char> G(55,"VANSH");

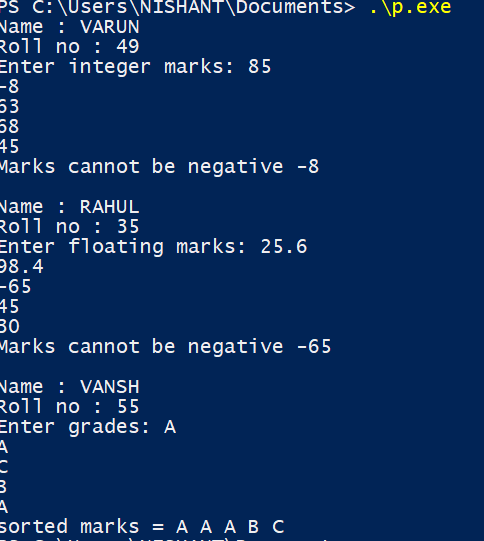
cout<<"Enter grades: ";

G.get\_marks();

return 0;

}





**25. Implementation of friend function and friend class**

#include <iostream>

#include <stdio.h>

using namespace std;

class Colony;

class Sector;

class containment

{

private:

int Capital;

int start;

int end;

int num;

public:

void cal(Sector z);

};

class Sector{

private:

int num;

string name;

int cnt;

string status;

public:

void get()

{

cout<<"Enter Sector number: ";

cin>>num;

cout<<"Enter Sector name: ";

cin>>name;

cout<<"Number of registered cases : ";

cin>>cnt;

}

void allocate(){

if(cnt > 100){

status = "red";

}else if(cnt > 50){

status = "orange";

}else{

status = "green";

}

}

friend void check(Sector z, Colony s);

friend void containment::cal(Sector z);

};

void containment::cal(Sector z){

Capital = z.cnt\*15000;

}

class Colony{

private:

string name;

int id;

string zs;

int tot;

int aff;

public:

void get(){

cout<<"Enter Colony name: ";

cin>>name;

cout<<"Enter Colony ID: ";

cin>>id;

cout<<"Enter Sector of Colony: ";

cin>>zs;

cout<<"Enter Total number of flats: ";

cin>>tot;

cout<<"Enter number of flats affected: ";

cin>>aff;

}

friend void check(Sector z, Colony s);

friend class containment;

};

void check(Sector z, Colony s){

if(s.tot < z.cnt/8){

cout<<"Colony is safe\n";

}else{

cout<<"Colony is unsafe\n";

}

}

int main(){

Sector a;

a.get();

a.allocate();

cout<<"\n";

Colony s;

s.get();

cout<<"\n";

check(a,s);

cout<<"\n";

containment c;

c.cal(a);

return 0;

}

