

Department of Information Technology

Delhi Technological University

OBJECT ORIENTED PROGRAMMING LAB (IT-203)



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

Submitted By:

Name : Varun Kumar

Roll No. : 2K19 / IT / 140

Group : G3

Submitted To:-

Mrs. Geetanjali Bhola

**Assistant Professor
Dept. of IT, DTU**

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>

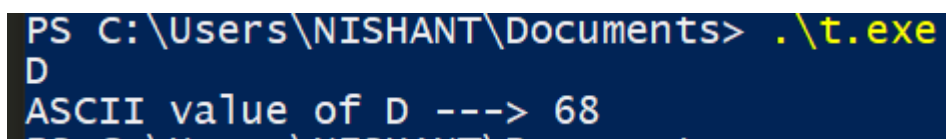
int main() {

    char c;

    scanf("%c", &c);

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

    return 0;
}
```



```
PS C:\Users\NISHANT\Documents> .\t.exe
D
ASCII value of D ---> 68
```

2. To print alternate characters of your name

```
#include <stdio.h>

#include <string.h>
```

```

int main() {

char myname[50];

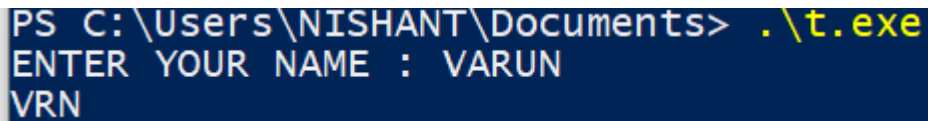
printf("ENTER YOUR NAME : ");
fgets(myname, sizeof(myname), stdin);
int len=strlen(myname);

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

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

}
return 0;
}

```



```

PS C:\Users\NISHANT\Documents> .\t.exe
ENTER YOUR NAME : VARUN
VRN

```

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;

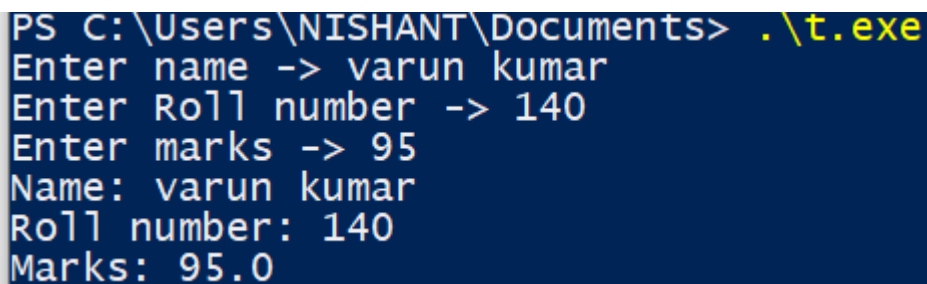
printf("Enter name -> ");
fgets(s.name, sizeof(s.name), stdin);

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;
}

```



```

PS C:\Users\NISHANT\Documents> .\t.exe
Enter name -> varun kumar
Enter Roll number -> 140
Enter marks -> 95
Name: varun kumar
Roll number: 140
Marks: 95.0

```

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 );
```

```

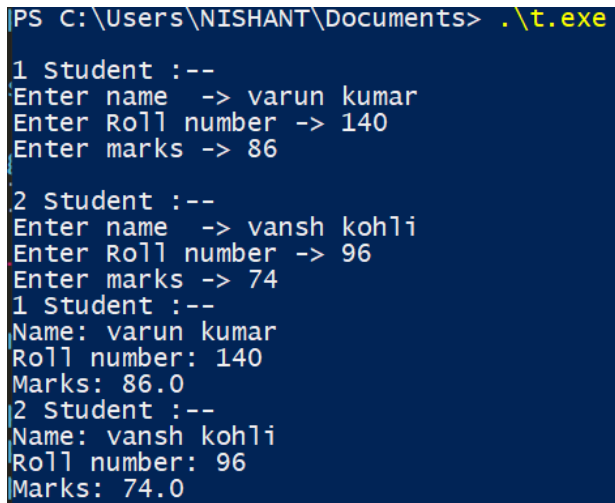
printf("Name: ");
printf("%s\n", s[i].name);
printf("Roll number: %d\n", s[i].roll);
printf("Marks: %.1f\n", s[i].marks);

}

return 0;

}

```



```

PS C:\Users\NISHANT\Documents> .\t.exe
1 Student :--
Enter name -> varun kumar
Enter Roll number -> 140
Enter marks -> 86
2 Student :--
Enter name -> vansh kohli
Enter Roll number -> 96
Enter marks -> 74
1 Student :--
Name: varun kumar
Roll number: 140
Marks: 86.0
2 Student :--
Name: vansh kohli
Roll number: 96
Marks: 74.0

```

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;
```

```
}
```

```
PS C:\Users\NISHANT\Documents> .\t.exe
```

```
1 Student :--  
Enter name -> varun kumar  
Enter Roll number -> 140  
Enter marks -> 86  
Enter HouseNo -> b68
```

```
Enter Area ->Ganesh Nagar
```

```
2 Student :--  
Enter name -> vansh kohli  
Enter Roll number -> 86  
Enter marks -> 74  
Enter HouseNo ->s70
```

```
Enter Area ->pandav nagar
```

```
1 Student :--  
Name: varun kumar  
Roll number: 140  
Marks: 86.0  
HouseNo : b68  
Area : Ganesh Nagar
```

```
2 Student :--  
Name: vansh kohli  
Roll number: 86  
Marks: 74.0  
HouseNo : s70  
Area : pandav nagar
```

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;
}
```

```
Name: Singh
employee ID: 1
salary: 50000
city of work : Delhi
total salary with bonuses : 70000

Name: Rawat
employee ID: 2
salary: 60000
city of work : Lucknow
total salary with bonuses : 70000

Name: Kaushik
employee ID: 3
salary: 50000
city of work : Chandigarh
total salary with bonuses : 65000

Name: Verma
employee ID: 4
salary: 55000
city of work : Agra
total salary with bonuses : 66000

Name: Sharma
employee ID: 5
salary: 40000
city of work : Amritsar
total salary with bonuses : 53000
```

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;
}
```

```

PS C:\Users\NISHANT\Documents> .\oop.exe
enter name: Singh
enter employee ID: 1
enter salary: 50000
enter city of work: Delhi
enter branch ID: 1
enter branch city: Delhi
enter bonus_offered: 20000
enter branch address:-
Enter Building_no : B-67
enter Area: Ganesh Nagar Complex
enter name: Rawat
enter employee ID: 2
enter salary: 60000
enter city of work: Chandigarh
enter branch ID: 20
enter branch city: Chandigarh
enter bonus_offered: 10000
enter branch address:-
Enter Building_no : A-87
enter Area: Mayur Vihar ph-2
enter name: Kaushik
enter employee ID: 3
enter salary: 50000
enter city of work: Lucknow
enter branch ID: 30
enter branch city: Lucknow
enter bonus_offered: 15000
enter branch address:-
Enter Building_no : F-105
enter Area: Pandav Nagar

enter employee id to be searched: 2

employee name: Rawat
Branch_address :-
Building_no : A-87
Area: Mayur Vihar ph-2
City : Chandigarh

```

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>>a;

Void print_data()//cout<<&&

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;
```

```
}
```

```

PS C:\Users\NISHANT\Documents> .\oop.exe
enter name: varun
enter age: 34
enter empid: 1
enter salary: 60000
enter experience: 2

enter name: rahul
enter age: 38
enter empid: 2
enter salary: 75000
enter experience: 5

name: varun
age: 34
empid: 1
salary: 60000
experience: 2
bonus: 4000

name: rahul
age: 38
empid: 2
salary: 75000
experience: 5
bonus: 10000

```

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> 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;
}

```

```

PS C:\Users\NISHANT\Documents> .\oop.exe
enter name: abhishek
enter age: 16
enter marks: 84

enter name: varun
enter age: 19
enter marks: 58

name: abhishek
group_allotted: A
status: N
grade: A

name: varun
group_allotted: B
status: E
grade: D

```


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; }
```

```
PS C:\Users\NISHANT\Documents> .\t.exe
Welcome - Enter name , roll , marks , Package_offered , Age , College_id:-
Varun
140
86
17
19
140
Enter name , College_id , State , Sem_fees_btech , Percentage_stud_placement:-
DTU
140
Delhi
200000
85
Tax calculated: 1600000
PS C:\Users\NISHANT\Documents>
```

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;
```

```
}
```

```
PS C:\Users\NISHANT\Documents> .\oop.exe
enter model_no , name and price :-
100
audi
800000
enter policy_no , name_of_policy , amount : -
5
car_insurance
200000
enter name , location , average_bill:-
CNG
Delhi
80000

model_no: 100
name: audi
price: 800000
insurance_expected: 8000

policy_no: 5
name_of_policy: car_insurance
amount: 200000
discount: 20000

name: CNG
location: Delhi
average_bill: 80000
discount: 8000

my_car_insurance: 8000
my_insur_discount: 20000
my_service_station_amount: 8000
total_expenditure: 36000
PS C:\Users\NISHANT\Documents>
```

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;
```

```
}
```

```
PS C:\Users\NISHANT\Documents> .\oop.exe  
parameterized with three parameters called  
default private constructor called  
DESTRUCTOR called and object destroyed  
parameterized with one parameter called  
copy constructor called  
parameterized with three parameters called  
default private constructor called  
DESTRUCTOR called and object destroyed  
varun 140 94  
DESTRUCTOR called and object destroyed  
DESTRUCTOR called and object destroyed  
DESTRUCTOR called and object destroyed  
DESTRUCTOR called and object destroyed
```

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 constructor 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;
```

```
}
```

```
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;
    error: invalid use of member 'employee::salary' in static member function  x
return bonus;    //ERROR
}

static int return_bonus(){ // static accessing static variable

cout<<"static member function called "<<endl;
return bonus;    // WORKS
}
```

ERROR WHILE STATIC MEMBER FUNCTION ACCESSING NON STATIC VARIABLES

```

PS C:\Users\NISHANT\Documents> .\oop.exe
default constructor called
default constructor called
default constructor called
Enter name , empid , salary:
Varun
5
200000
objects created: 3
static member function called
bonus_offered: 1000
default constructor called
Enter name , empid , salary:
rahul
25
400000
default constructor called
private constructor called
name: Varun
empid: 5
salary: 200000
PS C:\Users\NISHANT\Documents>

```

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; }
```



```

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;

```

error: invalid conversion from 'const student*' to 'student*' [-fpermissive] ×

error: assignment of read-only variable 'ptr1' ×

ERROR WHILE CONSTANT POINTER MODIFIES

```

PS C:\Users\NISHANT\Documents> .\oop.exe
DECLARED CONSTANT variable for STUDENT 1
DELHI TECHNOLOGICAL UNIVERSITY
DECLARED CONSTANT variable for STUDENT 2
DELHI TECHNOLOGICAL UNIVERSITY
PS C:\Users\NISHANT\Documents>

```

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,"VANSI"),student(100,"RAHUL"),student(125,"PI
YUSH"),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;
}

```

```

PS C:\Users\NISHANT\Documents> .\oop.exe
roll_no:      122
NAME:  VANSI
roll_no:      140
NAME:  varun
roll_no:      125
NAME:  PIYUSH
roll_no:      100
NAME:  RAHUL
roll_no:      151
NAME:  VEDANT

```

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;
}
```

```

PS C:\Users\NISHANT\Documents> .\p.exe
Enter the employee number : 1
Enter the salary : 50000
choose your designation : 1.Asst manager || 2.clerk || 3.vp
1
Enter the employee number : 2
Enter the salary : 80000
choose your designation : 1.Asst manager || 2.clerk || 3.vp
1
Enter the employee number : 3
Enter the salary : 30000
choose your designation : 1.Asst manager || 2.clerk || 3.vp
2
Enter the employee number : 4
Enter the salary : 20000
choose your designation : 1.Asst manager || 2.clerk || 3.vp
2
Enter the employee number : 5
Enter the salary : 120000
choose your designation : 1.Asst manager || 2.clerk || 3.vp
3
Enter the employee number : 6
Enter the salary : 150000
choose your designation : 1.Asst manager || 2.clerk || 3.vp
3

Avg salary for Asst manager is 65000
Avg salary for clerk is 25000
Avg salary for vp is 135000

```

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
```

```
{
```

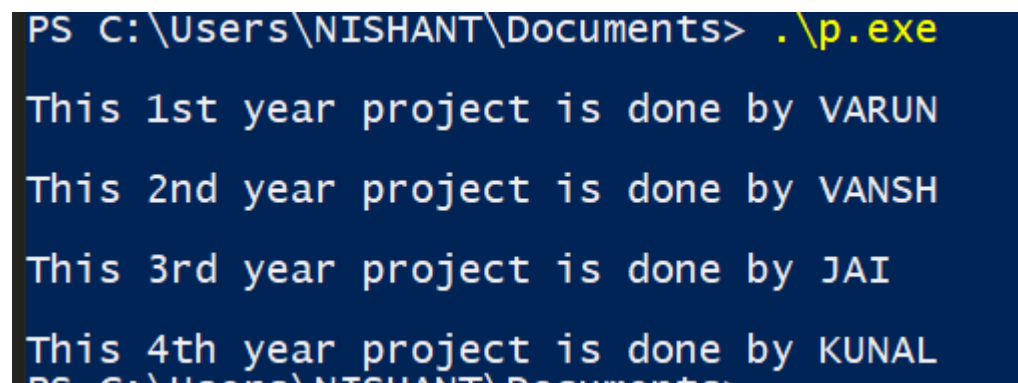


```

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

int main()
{
    Student s1("VARUN");
    Student s2("VANSI");
    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);
}

```



```

PS C:\Users\NISHANT\Documents> .\p.exe
This 1st year project is done by VARUN
This 2nd year project is done by VANSI
This 3rd year project is done by JAI
This 4th year project is done by KUNAL
PS C:\Users\NISHANT\Documents>

```

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("VANSI", 24);
    employee * e2 = new employee("RAHUL",25);

    e1->display();
    e2->display();

    delete e1;
    delete e2;
}

```

```
}
```

```
PS C:\Users\NISHANT\Documents> .\p.exe
overloading new operator with size: 28
Constructor is called
overloading new operator with size: 28
Constructor is called
Name:VANSH
Age:24
Name:RAHUL
Age:25
overloading delete operator
overloading delete operator
```

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;
}

```

```

PS C:\Users\NISHANT\Documents> .\p.exe
Enter the value of object : 15
35
First Distance: Feet : 15 Inches : 5
Second Distance: Feet : 8 Inches : 22
Third Distance: Feet : 15 Inches : 35

Now overloading = operator:
First Distance: Feet : 8 Inches : 22
PS C:\Users\NISHANT\Documents>

```

OVERLOADING [] OPERATOR

```

#include <iostream>
using namespace std;
const int SIZE = 10;

class safearray {
private:
    int arr[SIZE];

public:
    safearray() {
        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() {
    safearray A;

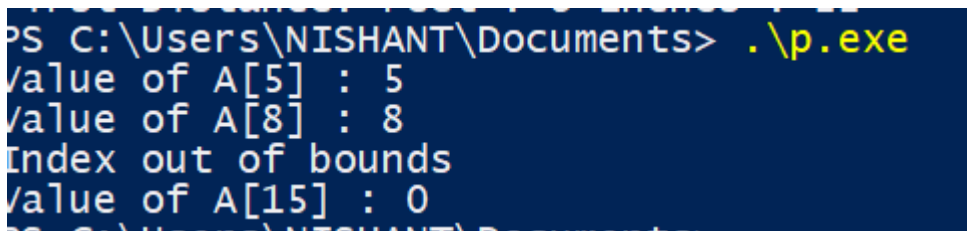
```

```

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;
}

```



```

PS C:\Users\NISHANT\Documents> .\p.exe
Value of A[5] : 5
Value of A[8] : 8
Index out of bounds
Value of A[15] : 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 college_student , protected college_employee{
```

```
string college_name;
```

```
public:
```

```
college_people(string na,int r,string n,int e,int s ,string c):  
college_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");
```

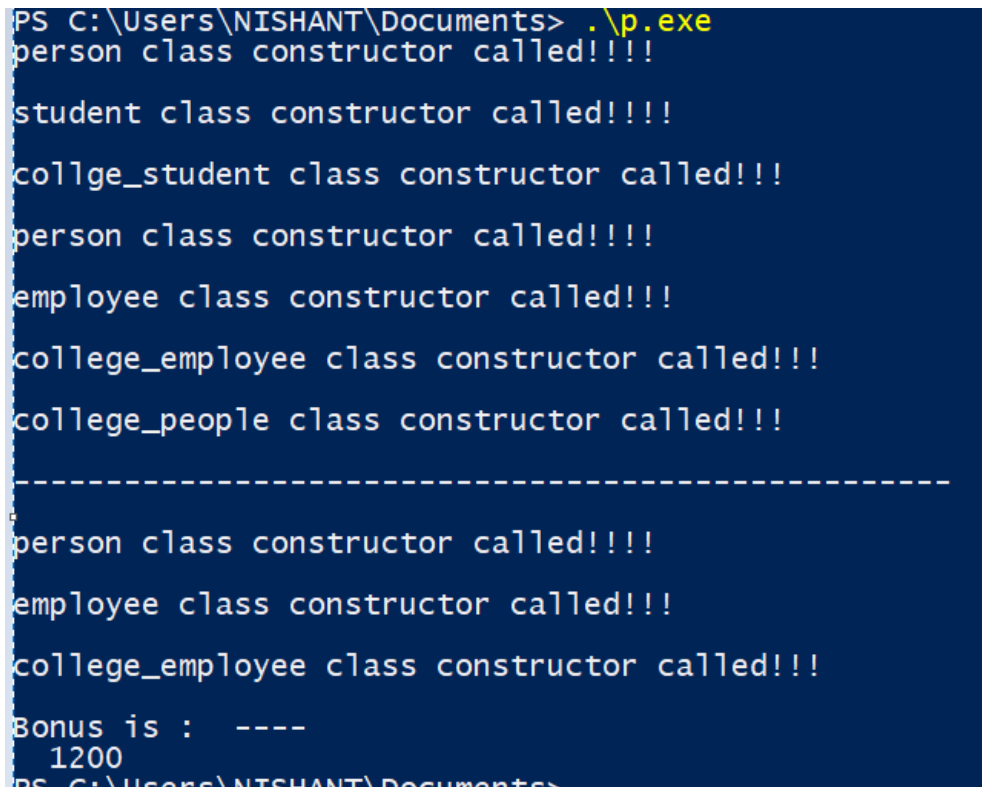
```
cout<<"-----\n\n";
```

```
college_employee E("rahul",694,60000);
```

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

```
return 0;
```

```
}
```



```
PS C:\Users\NISHANT\Documents> .\p.exe
person class constructor called!!!!
student class constructor called!!!!
college_student class constructor called!!!
person class constructor called!!!!
employee class constructor called!!!
college_employee class constructor called!!!
college_people class constructor called!!!
-----
person class constructor called!!!!
employee class constructor called!!!
college_employee class constructor called!!!
Bonus is : ----
1200
PS C:\Users\NISHANT\Documents>
```

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;
}

```

```

PS C:\Users\NISHANT\Documents> .\m.exe
Animal class constructor called !!!
Herbivores class constructor called !!!
Carnivores class constructor called !!!
Omnivores class constructor called !!!
Omnivores class destructor called !!!
Carnivores class destructor called !!!
Herbivores class destructor called !!!
Animal class destructor called !!!

```

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\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\tList of Documents Required\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\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\tList of Documents Required\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\tEnter your choice:";

        cin >>x;

        switch(x)

        {

            case 1:

            {

                int a;

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

                do

                {

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

                    cout<<"\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"<<c1->name<<endl<<endl;
    do
    {

```

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

```

PS C:\Users\NISHANT\Documents> .\m.exe
1.School Admission          2.College Admission          3.Exit
                             Enter your choice:2

                             Delhi Technological University, Delhi

1.Procedure Details        2.Documents Required        3.Contact          4.Exit
                             Enter your choice:1
                             Steps to follow

Step 1:
Fee Payment

Step 2:
Registration

Step 3:
Login to JAC Delhi Account

Step 4:
Choice Filling

Step 5:
Choice Locking and Printing of Locked Choices

Step 6:
Reporting at Allotted Institute

1.Procedure Details        2.Documents Required        3.Contact          4.Exit
                             Enter your choice:2
                             List of Documents Required
1.Applicable fee payment receipt of Rs. 10,000
2.Three passport size photographse
3.Self-attested copies of Admit Card and Score Card of JEE Main 2020
4.Original marksheet of the qualifying examination i.e. Class 12th
5.Original Date of Birth certificate as indicated in High School examination i.e. Class 10th
6.Photo Identity proof of both the parents
7.Photocopy of Residential Address Proof

1.Procedure Details        2.Documents Required        3.Contact          4.Exit
                             Enter your choice:4
Thank You for visiting Delhi Technological University, Delhi

Thanks for visiting
1.School Admission          2.College Admission          3.Exit
                             Enter your choice:3

Thanks for visiting

```

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\tWITH VIRTUAL DESTRUCTOR\n";
    A *ob1=new B;
    cout<<endl;
    delete ob1;

    cout<<endl;
    cout<<"\t\tWITHOUT VIRTUAL DESTRUCTOR\n";
    C* ob2=new D;
    cout<<endl;
    delete ob2;
    return 0;
}

```



```

PS C:\Users\NISHANT\Documents> .\m.exe
                                WITH VIRTUAL DESTRUCTOR
Constructor of A
Constructor of B

Destructor of B
Destructor of A

                                WITHOUT VIRTUAL DESTRUCTOR
Constructor of C
Constructor of D

Destructor of C

```

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,"VANSI");  
    cout<<"Enter grades: ";  
    G.get_marks();  
    return 0;  
}
```

```

PS C:\Users\NISHANT\Documents> .\p.exe
Name : VARUN
Roll no : 49
Enter integer marks: 85
96
45
88
56
sorted marks = 45 56 85 88 96

Name : RAHUL
Roll no : 35
Enter floating marks: 65.5
74.9
85.64
96
56.4
sorted marks = 56.4 65.5 74.9 85.64 96

Name : VANSI
Roll no : 55
Enter grades: A
C
B
B
A
sorted marks = A A B B C
PS C:\Users\NISHANT\Documents>
Name : RAHUL
Roll no : 35
Enter floating marks: 25.6
98.4
-65
45
30
Marks cannot be negative -65

Name : VANSI
Roll no : 55
Enter grades: A
A
C
B
A
sorted marks = A A A B C

```

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;  
}
```

```
PS C:\Users\NISHANT\Documents> .\p.exe
Enter Sector number: 45
Enter Sector name: noida
Number of registered cases : 500

Enter Colony name: GANESH
Enter Colony ID: 14
Enter Sector of Colony: 36
Enter Total number of flats: 100
Enter number of flats affected: 60

Colony is unsafe
```