|  |  |
| --- | --- |
| **Assignment No.** | 1 (GROUP A) |
| **Title** | Arithmetic operations on complex numbers using operator overloading. |
| **Subject** | Object Oriented Programming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 1**

#include<iostream>

using namespace std;

class complex

{

float real;

float imag;

public:

complex();

complex(float,float);

complex operator+(complex);

complex operator\*(complex);

friend ostream &operator<<(ostream &out, complex c);

friend istream &operator>>(istream &in, complex &c);

};

complex::complex()

{

real=0;

imag=0;

}

complex::complex(float a,float b)

{

real=a;

imag=b;

}

complex complex::operator+(complex c)

{

complex temp;

temp.real=real+c.real;

temp.imag=imag+c.imag;

return temp;

}

complex complex::operator\*(complex c)

{

complex temp2;

temp2.real=(real\*c.real)-(imag\*c.imag);

temp2.imag=(imag\*c.real)+(real\*c.imag);

return temp2;

}

ostream &operator<<(ostream &out, complex c) //output

{

out<<c.real<<"+"<<c.imag<<"i"<<"\n";

return out;

}

istream &operator>>(istream &in, complex &c) //input

{

cout<<"enter real part:\n";

in>>c.real;

cout<<"enter imag part: \n";

in>>c.imag;

return in;

}

/\*complex complex::operator /(complex c)

{

complex temp3;

temp3.x=((x\*c.x)+(y\*c.y))/((c.x\*c.x)+(c.y\*c.y));

temp3.y=((y\*c.x)-(x\*c.y))/((c.x\*c.x)+(c.y\*c.y));

return(temp3);

}

\*/

int main()

{

complex c1,c2,c3;

cout<<"\n Enter the 1st complex number\n";

cin>>c1;

cout<<"\n Enter the 2nd complex number\n";

cin>>c2;

cout<<"\n The first number complex is ";

cout<<c1;

cout<<"\n";

cout<<"\n The second complex number is ";

cout<<c2;

cout<<"\n";

c3=c1+c2;

cout<<"Addition of complex number=";

cout<<c3;

cout<<"\n";

c3=c1\*c2;

cout<<"Multiplication of complex number=";

cout<<c3;

cout<<"\n";

return 0;

}

|  |  |
| --- | --- |
| **Assignment No.** | 2 (GROUP A) |
| **Title** | Personnel information system using constructor, destructor, static member functions, friend class, this pointer, inline code and dynamic memory allocation. |
| **Subject** | Object Oriented Programming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 2**

#include <iostream>

using namespace std;

class info

{

int roll\_no;

public:

info()

{

roll\_no=1;

}

~info()

{

cout<<"\ndestructor called from info class\n";

}

info(info &i)

{

this->roll\_no=i.roll\_no;

}

friend class student;

};

class student

{

string name;

public:

student()

{

name="Jishu";

}

~student()

{

cout<<"\ndestructor called from student class\n";

}

student(student &s)

{

this->name=s.name;

}

void create(info &i)

{

cout<<"\nEnter your name";

cin>>name;

cout<<"Enter your roll number\n";

cin>>i.roll\_no;

}

void display(info &i)

{

cout<<"\nYour name "<<name<<endl;

cout<<"\nYour roll number "<<i.roll\_no<<endl;

}

static int cnt;

inline static void objcount()

{

cnt++;

}

inline static void objcountshow()

{

cout<<cnt;

}

};

int student::cnt;

int main()

{

cout<<"\nDefault constructor\n";

info in1;

student st1;

st1.display(in1);

cout<<"+++++++++++++++++++++++++++++++++++++";

cout<<"\nCreate database and Display database";

int n;

cout<<"\nEnter number of students details\n";

cin>>n;

info \*i1=new info[n];

student \*s1=new student[n];

cout<<"\nCreate database\n";

for(int k=0;k<n;k++)

{

s1[k].create(i1[k]);

student::objcount();

}

cout<<"+++++++++++++++++++++++++++++++++++++";

cout<<"\nTotal number of object created ";

student::objcountshow();

cout<<endl<<"+++++++++++++++++++++++++++++++++++++";

cout<<"\nDisplay database\n";

for(int k=0;k<n;k++)

{

s1[k].display(i1[k]);

}

cout<<"+++++++++++++++++++++++++++++++++++++";

cout<<"\nCopy constructor\n";

info in2;

student st2;

st2.create(in2);

student st3(st2);

info in3(in2);

cout<<"+++++++++++++++++++++++++++++++++++++";

cout<<"\nCopy constructor\n";

st3.display(in3);

cout<<"+++++++++++++++++++++++++++++++++++++";

return 0;

}

|  |  |
| --- | --- |
| **Assignment No.** | 3 (GROUP A) |
| **Title** | Creating a class which uses the concept of inheritance, displays data and data members and uses the concept of exception handling. |
| **Subject** | Object Oriented Programming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 3**

#include<string>

#include<iostream>

using namespace std;

class publication

{

public:

string title;

float price;

void getdata()

{

cout<<"enter title"<<"\t";

cin>>title;

if(title.length()<=0)

throw title;

cout<<"enter price:"<<"\t";

cin>>price;

if(price<=0.0)

throw price;

}

};

class book:public publication

{

public:

int page\_count;

void getPageCount()

{

try

{

getdata();

cout<<"\nenter page conut";

cin>>page\_count;

if(page\_count<0)

throw page\_count;

else

display();

}

catch(...)

{

cout<<"invalid Data\n";

setdefault();

display();

}

}

void display()

{

cout<<"Book details\n";

cout<<"title is:"<<title<<"\t";

cout<<"price is:"<<price<<"\t";

cout<<"page count="<<page\_count<<endl;

}

void setdefault()

{

title="0";

price=0;

page\_count=0;

}

};

class tape:public publication

{

public:

float play\_time;

void getPlayTime()

{

try

{

getdata();

cout<<"enter play time in minute";

cin>>play\_time;

if(play\_time<0.0)

throw play\_time;

else

display();

}

catch(...)

{

cout<<"invalid Data\n";

setdefault();

display();

}

}

void display()

{

cout<<"Audio caset details\n";

cout<<"title is:"<<title<<"\t";

cout<<"price is:"<<price<<"\t";

cout<<"play\_time="<<play\_time<<endl;

}

void setdefault()

{

title="0";

price=0;

play\_time=0;

}

};

int main()

{

book b;

tape t;

b.getPageCount();

t.getPlayTime();

return 0;

}

|  |  |
| --- | --- |
| **Assignment No.** | 4(GROUP B) |
| **Title** | Write a C++ program that creates an output file, writes information  to it, closes the file and open it again as an input file and read  the information from the file. |
| **Subject** | Object OrientedProgramming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 4**

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

char name[30];

float cost;

cout<<"enter item name";

cin>>name;

cout<<"enter item cost";

cin>>cost;

ofstream outf("item");

outf<<"name ="<<name;

outf<<"\ncost ="<<cost;

outf.close();

ifstream inf("item");

inf.open("item");

inf>>name;

inf>>cost;

cout<<"Item name is: = " <<name;

cout<<"Item cost is: = " <<cost;

inf.close();

return 0;

}

|  |  |
| --- | --- |
| **Assignment No.** | 5(GROUP B) |
| **Title** | Write a function template selection sort. Write a program that inputs, sorts and outputs an integer array and a float array. |
| **Subject** | Object Oriented Programming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 5**

#include<iostream>

using namespace std;

int n;

template<class f>

void sel(f A[10])

{

int i,j,min;

f temp;

for(i=0;i<n-1;i++)

{

min=i;

for(j=i+1;j<n;j++)

{

if(A[j]<A[min])

min=j;

}

temp=A[i];

A[i]=A[min];

A[min]=temp;

}

cout<<"\nSorted array:";

for(i=0;i<n;i++)

{

cout<<" "<<A[i];

}

}

int main()

{

int A[10];

float B[10];

int i;

cout<<"\nEnter total no of int elements:";

cin>>n;

cout<<"\nEnter int elements:";

for(i=0;i<n;i++)

{

cin>>A[i];

}

sel<int>(A);

cout<<"\nEnter total no of float elements:";

cin>>n;

cout<<"\nEnter float elements:";

for(i=0;i<n;i++)

{

cin>>B[i];

}

sel<float>(B);

}

|  |  |
| --- | --- |
| **Assignment No.** | 6(GROUP C) |
| **Title** | Write C++ program using STL for sorting and searching user  defined records such as Item records (Item code, name, cost, quantity etc) using vector container. |
| **Subject** | Object Oriented Programming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 6**

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

class Item

{

public:

char name[10];

int quantity;

int cost;

int code;

bool operator==(const Item& i1)

{

return code==i1.code;

}

};

vector<Item> o1;

void print(Item &i1);

void display();

void insert();

void search();

void dlt();

bool compare(const Item &i1, const Item &i2)

{

return i1.cost < i2.cost;

}

int main()

{

int ch;

do

{

cout<<"\n\*\* Menu \*\*";

cout<<"\n1.Insert";

cout<<"\n2.Display";

cout<<"\n3.Search";

cout<<"\n4.Sort";

cout<<"\n5.Delete";

cout<<"\n6.Exit";

cout<<"\nEnter your choice:";

cin>>ch;

switch(ch)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

sort(o1.begin(),o1.end(),compare);

cout<<"\n\n Sorted on Cost";

display();

break;

case 5:

dlt();

break;

case 6:

exit(0);

}

}while(ch!=7);

return 0;

}

void insert()

{

Item i1;

cout<<"\nEnter Item Name:";

cin>>i1.name;

cout<<"\nEnter Item Quantity:";

cin>>i1.quantity;

cout<<"\nEnter Item Cost:";

cin>>i1.cost;

cout<<"\nEnter Item Code:";

cin>>i1.code;

o1.push\_back(i1);

}

void display()

{

for\_each(o1.begin(),o1.end(),print);

}

void print(Item &i1)

{

cout<<"\n";

cout<<"\nItem Name:"<<i1.name;

cout<<"\nItem Quantity:"<<i1.quantity;

cout<<"\nItem Cost:"<<i1.cost;

cout<<"\nItem Code:"<<i1.code;

}

void search()

{

Item i1;

cout<<"\nEnter Item Code to search:";

cin>>i1.code;

vector<Item>::iterator p;

p=find(o1.begin(),o1.end(),i1);

if(p==o1.end())

{

cout<<"\nNot found.";

}

else

{

cout<<"\nFound."<<endl;

cout<<"Item Name : "<<p ->name<<endl;

cout<<"Item Quantity : "<<p ->quantity<<endl;

cout<<"Item Cost : "<<p ->cost<<endl;

cout<<"Item Code: "<<p ->code<<endl;

}

}

void dlt()

{

vector<Item>::iterator p;

Item i1;

cout<<"\nEnter Item Code to delete:";

cin>>i1.code;

p=find(o1.begin(),o1.end(),i1);

if(p==o1.end())

{

cout<<"\nNot found.";

}

else

{

o1.erase(p);

cout<<"\nDeleted.";

}

}

|  |  |
| --- | --- |
| **Assignment No.** | 7(GROUP C) |
| **Title** | Write a program in C++ to use map associative container. The keys will be the names of states and the values will be the populations of the states. When the program runs, the user is prompted to type the name of a state. The program then looks in the map, using the state name as an index and returns the  population of the state |
| **Subject** | Object Oriented Programming |
| **Class** | S.E. (C.E.) |
| **Roll No.** |  |
| **Date** |  |
| **Signature** |  |

**Lab Assignment 7**

#include<iostream>

#include<map>

#include<string>

using namespace std;

int main()

{

typedef map<string,int> mapType;

mapType populationMap;

populationMap.insert(pair<string, int>("Maharashtra", 7026357));

populationMap.insert(pair<string, int>("Rajasthan", 6578936));

populationMap.insert(pair<string, int>("Karanataka", 6678993));

populationMap.insert(pair<string, int>("Punjab", 5789032));

populationMap.insert(pair<string, int>("West Bengal", 6676291));

mapType::iterator iter;

cout<<"========Population of states in India==========\n";

cout<<"\n Size of populationMap"<<populationMap.size()<<"\n";

string state\_name;

cout<<"\n Enter name of the state :";

cin>>state\_name;

iter = populationMap.find(state\_name);

if( iter!= populationMap.end() )

cout<<state\_name<<" 's population is "

<<iter->second ;

else

cout<<"Key is not populationMap"<<"\n";

populationMap.clear();

}