**Lab Cycle 1**  
// swapping using call by value and call by reference//

#include<iostream>

using namespace std;

int swap1(int a, int b);

int swap2(int &a,int &b);

int main()

{

int a,b;

cout<<"Enter values of a,b: \n";

cin>>a>>b;

swap1(a,b);

swap2(a,b);

return 0;

}

int swap1(int a,int b)

{

cout<<"values of a, b before swapping are "<<a<<", "<<b<<"\n";

a=a+b;

b=a-b;

a=a-b;

cout<<"After swapping a,b values are "<<a<<", "<<b<<endl;

return 0;

}

int swap2(int &a, int &b)

{

cout<<"values of a, b before swapping are "<<a<<", "<<b<<endl;

a=a+b;

b=a-b;

a=a-b;

cout<<"After swapping a,b values are "<<a<<", "<<b<<endl;

return 0;

}

####output

Enter values of a,b:

5 8

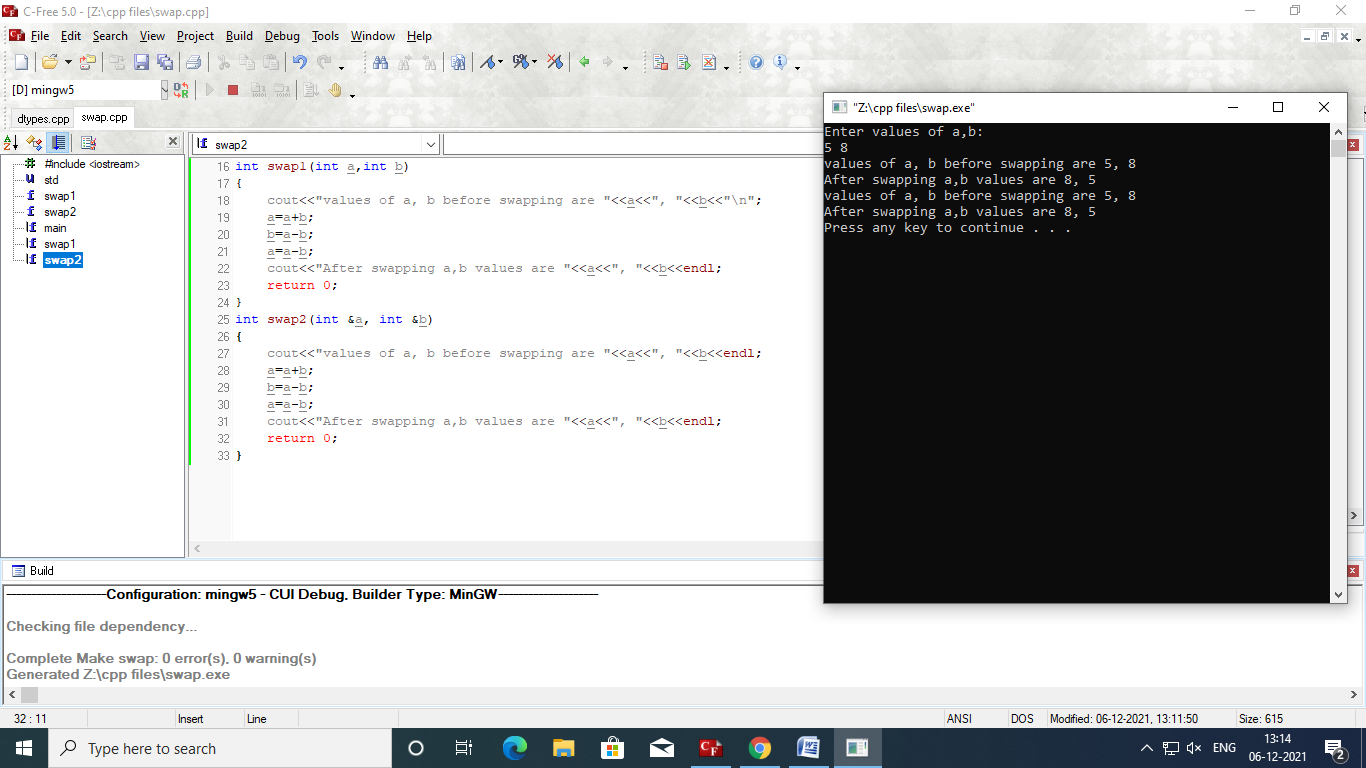
values of a, b before swapping are 5, 8

After swapping a,b values are 8, 5

values of a, b before swapping are 5, 8

After swapping a,b values are 8, 5

Press any key to continue . . .

**

**lab cycle2**

#include<iostream>  
#include<cstring>  
using namespace std;  
void abcd(char \*x,char \*y);  
void abcd(char \*str1,char \*str2,int l);  
int main()  
{  
      char st1[20]="Heyy",st2[20]="Dushyanth",st3[40]="RVR&JC",st4[40]="College Of Engineering";  
      abcd(st1,st2);  
      cout<<st1<<endl;  
      abcd(st3,st4,7);  
      cout<<st3<<endl;  
      return 0;  
}  
void abcd(char \*x,char \*y)  
{  
      strcat(x,y);  
}  
void abcd(char \*str1,char \*str2,int l)  
{  
      strncat(str1,str2,l);  
      cout<<"\n";  
}

**Output:**

HeyyDushyanth

RVR&JCCollege

Press any key to continue . . .

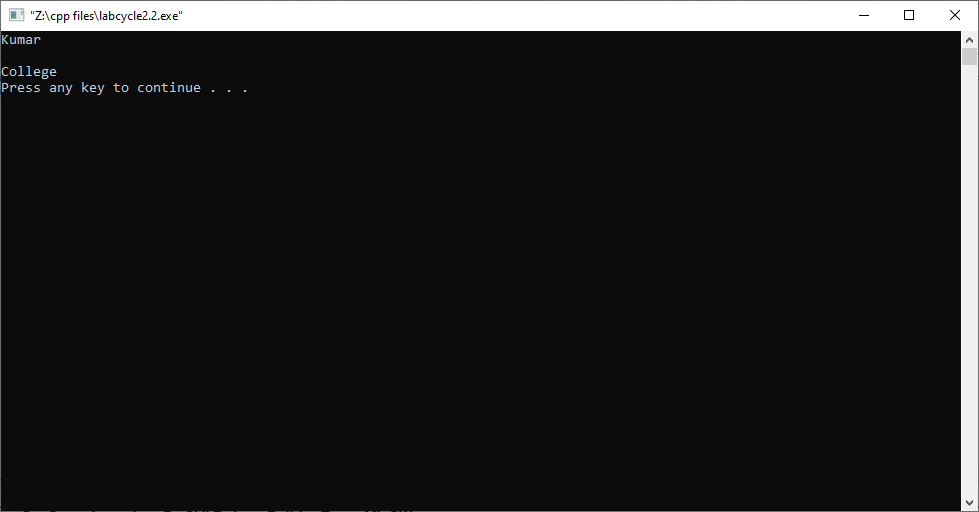
#include<iostream>  
#include<cstring>  
using namespace std;  
void abcd(char \*x,char \*y);  
void abcd(char \*str1,char \*str2,int l);  
int main(){  
      char str1[20]="Dushyanth ",str2[20]="Kumar ",str3[50]="RVR&JC",str4[40]="College Of Engineering";  
      abcd(str1,str2);  
      cout<<str1<<endl;  
      abcd(str3,str4,7);  
      cout<<str3<<endl;  
      return 0;  
}  
void abcd(char \*x,char \*y)  
{  
      strcpy(x,y);  
}  
void abcd(char \*str1,char \*str2,int l)  
{  
      strncpy(str1,str2,l);  
      cout<<"\n";  
}

**Output**

Kumar

College

Press any key to continue . . .



**lab cycle3**

#include<iostream>  
using namespace std;  
int main()  
{  
int \*ptr1=NULL;  
ptr1=new int;  
\*ptr1=28;  
float \*ptr2=new float(23.8);  
int \*ptr3=new int[20];  
cout<<"value of ptr1"<<"\t"<<\*ptr1<<endl;  
cout<<"value of ptr2"<<"\t"<<\*ptr2<<endl;  
cout<<"address of ptr1"<<ptr1<<endl;  
cout<<"address of ptr2"<<ptr2<<endl;  
if(!ptr3)  
cout<<"failed to allocate"<<endl;  
else  
{  
for(int i=10;i<15;i++)  
ptr3[i]=i+1;  
cout<<"values of blocks"<<endl;  
for(int i=10;i<15;i++)  
cout<<ptr3[i]<<endl;;  
}  
delete ptr1;  
delete ptr2;  
delete [] ptr3;  
cout<<\*ptr1;  
return 0;  
}

**Output:**

value of ptr1 23

value of ptr2 23.8

address of ptr10x700d28

address of ptr20x700da8

values of blocks

11

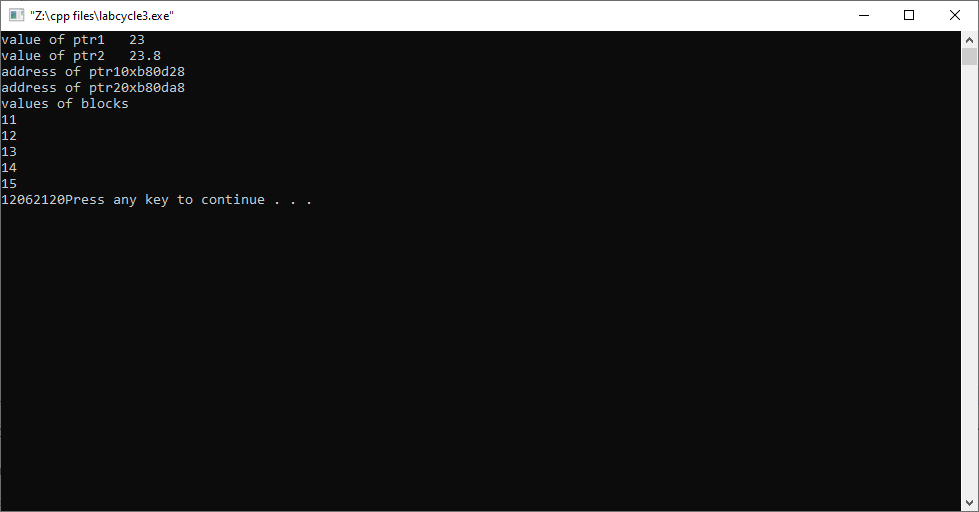
12

13

14

15

7343528Press any key to continue . . .



**lab cycle4**

#include<iostream>  
using namespace std;  
class Complex  
{  
int \*r,\*im;  
public:  
Complex(int \*a,int \*b)  
{  
 r=a;  
im=b;  
}  
Complex()  
{  
  
}  
Complex(Complex &c)  
{  
this->r=c.r;  
this->im=[c.im](http://c.im/);  
}  
~Complex()  
{  
  
cout<<"object destroyed"<<endl;  
}  
void display()  
{  
cout<<"number is ";  
cout<<\*r<<" "<<\*im<<endl;  
}  
void add(Complex a, Complex b)  
{  
Complex c3;  
int r1=\*a.r+\*b.r;  
int r2 =\*[a.im](http://a.im/)+\*[b.im](http://b.im/);  
c3.r=&r1;  
[c3.im](http://c3.im/)=&r2;  
c3.display();  
}  
void sub(Complex a, Complex b)  
{  
Complex c5;  
int r3=\*a.r-\*b.r;  
int r4=\*a.im-\*[b.im](http://b.im/);  
c5.r=&r3;  
[c5.im](http://c5.im/)=&r4;  
c5.display();  
}  
};  
int main()  
{  
int a=1,b=2;  
Complex c1(&a,&b);  
int p=3,q=4;  
Complex c2(&p,&q);  
c1.display();  
c2.display();  
Complex c3;  
c3.add(c1,c2);  
Complex c4(c1);  
c4.display();  
Complex c6(c2);  
c6.display();  
Complex c5;  
c5.sub(c1,c6);  
c2=c1;  
c2.display();  
return 0;  
}

**Output:**

number is 1 2

number is 3 4

number is 4 6

object destroyed

object destroyed

object destroyed

number is 1 2

number is 3 4

number is -2 -2

object destroyed

object destroyed

object destroyed

number is 1 2

object destroyed

object destroyed

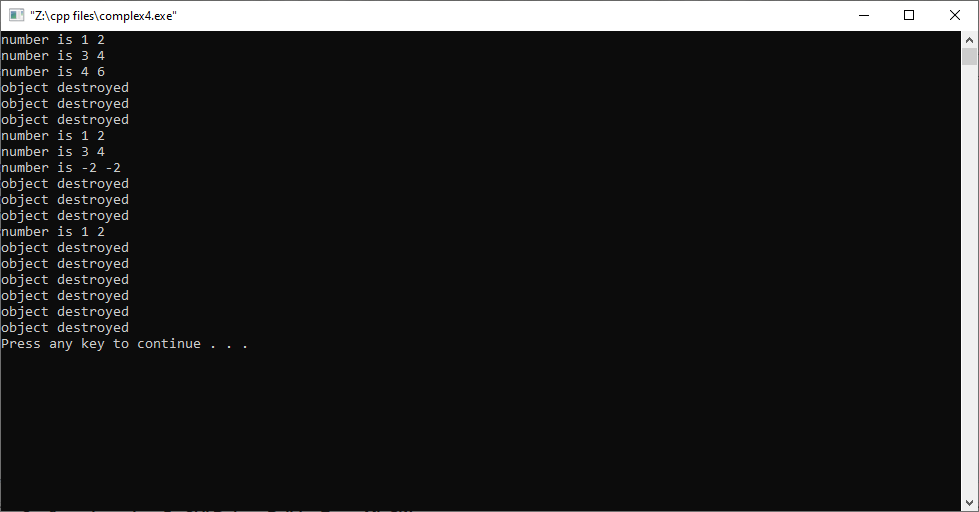
object destroyed

object destroyed

object destroyed

object destroyed

Press any key to continue . . .



**LAB CYCLE 5**

#include<iostream>

using namespace std;

class vector

{

int \*v,\*v1;

int n;

public:

vector()

{

v=(int\*)malloc(5\*sizeof(int));

cout<<"enter size of array";

cin>>n;

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

{

cin>>v[i];

}

}

void disp()

{

cout<<"accessing elements"<<endl;

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

{

cout<<v[i]<<endl;

}

}

~vector()

{

}

vector operator +=(int a)

{

cout<<"after using assignment operator"<<endl;

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

v[i]=v[i]+a;

return \*this;

}

void add()

{

v=(int\*)realloc(v,5\*sizeof(int));

for(int i=n;i<7;i++)

{

cout<<"enter new elements:";

cin>>v[i];

}

cout<<"after adding new elements"<<endl;

for(int i=0;i<7;i++)

{

cout<<v[i]<<endl;

}

}

void del()

{

int p;

cout<<"enter a position to delete:";

cin>>p;

if(p>n)

{

cout<<"deletion not possible"<<endl;

}

else

{

for(int j=p-1;j<n-1;j++)

{

v[j]=v[j+1];

}

cout<<"after deletion"<<endl;

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

{

cout<<v[i]<<endl;

}

}

}

};

int main()

{

vector c1,c2;

vector c3(c2);

c1.disp();

c2.disp();

c3.disp();

c1+=3;

c1.disp();

c2.add();

c1.del();

}

**Output:**

enter size of array3

1

11

111

enter size of array3

2

22

222

accessing elements

1

11

111

accessing elements

2

22

222

accessing elements

2

22

222

after using assignment operator

accessing elements

4

14

114

enter new elements:5

enter new elements:5

enter new elements:5

enter new elements:5

after adding new elements

2

22

222

5

5

5

5

enter a position to delete:3

after deletion

4

14

Press any key to continue . . .

**LAB CYCLE 6**

#include<iostream>

using namespace std;

class matrix

{

int v[5][5];

int a,b;

public:

matrix()

{

}

matrix(int n)

{

cout<<"enter values into array:"<<endl;

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

{

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

{

cin>>v[i][j];

}

}

}

~matrix()

{

}

void display()

{

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

{

cout<<"\n";

for(int j=0;j<3;j++)

{

cout<<v[i][j]<<"\t";

}

}

cout<<"\n";

}

void add(matrix a,matrix b)

{

cout<<"Addition of matrices"<<endl;

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

{

for(int j=0;j<3;j++)

{

a.v[i][j]=a.v[i][j]+b.v[i][j];

}

}

a.display();

}

void mul(matrix a,matrix b)

{

matrix c;

cout<<"Multiplication of matrices"<<endl;

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

{

for(int j=0;j<3;j++)

{

c.v[i][j]=0;

for(int k=0;k<3;k++)

{

c.v[i][j]+=a.v[i][k]\*b.v[k][j];

}

}

}

c.display();

}

};

int main()

{

matrix m1(3),m2(3);

m1.display();

m2.display();

matrix m3;

m1.add(m1,m2);

matrix m4;

m1.mul(m1,m2);

matrix m5(m1);

m5.display();

return 0;

}

**Output:**

enter values into array:

1

2

3

1

2

3

1

2

3

enter values into array:

3

2

1

3

2

1

3

2

1

1 2 3

1 2 3

1 2 3

3 2 1

3 2 1

3 2 1

Addition of matrices

4 4 4

4 4 4

4 4 4

Multiplication of matrices

18 12 6

18 12 6

18 12 6

1 2 3

1 2 3

1 2 3

Press any key to continue . . .

***Lab Cycle 8***

8. Define class stack, queue, linked-list, array, set using some data-type (int) with data members kept as private and functions kept in both protected and public sections.

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node()

{

data = 0;

next = NULL;

}

Node(int data)

{

this->data = data;

this->next = NULL;

}

};

class Linkedlist {

Node\* head;

public:

Linkedlist()

{

head = NULL;

}

void insertNode(int);

void printList();

void deleteNode(int);

};

void Linkedlist::insertNode(int data)

{

Node\* newNode = new Node(data);

if (head == NULL)

{

head = newNode;

return;

}

Node\* temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = newNode;

}

void Linkedlist::deleteNode(int del)

{

Node \*temp1 = head, \*temp2 = NULL;

int len=0;

if (head == NULL) {

cout << "List empty." << endl;

return;

}

while (temp1 != NULL) {

temp1 = temp1->next;

len++;

}

if (len<del) {

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

return;

}

temp1 = head;

if (del== 1) {

head = head->next;

delete temp1;

return;

}

while (del-- > 1) {

temp2 = temp1;

temp1 = temp1->next;

}

temp2->next = temp1->next;

delete temp1;

}

void Linkedlist::printList()

{

Node\* temp = head;

if (head == NULL) {

cout << "List empty" << endl;

return;

}

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

}

class stack

{

int stack[100];

static int n;

static int top;

public:

void push(int val) {

if(top >= n-1)

cout<<"Stack Overflow"<<endl;

else {

top++;

stack[top] = val;

}

}

void pop() {

if(top <= -1)

cout<<"Stack Underflow"<<endl;

else {

cout<<"The popped element is "<< stack[top] <<endl;

top--;

}

}

void display() {

if(top>= 0) {

cout<<"Stack elements are:";

for(int i = top; i>= 0; i--)

cout<<stack[i]<<" ";

cout<<endl;

} else

cout<<"Stack is empty";

}

};

int stack::n=20;

int stack::top=-1;

class queue

{

int queue[100];

static int n,front,rear;

public:

void enqueue() {

int val;

if (rear == n - 1)

cout<<"Queue Overflow"<<endl;

else {

if (front == - 1)

front = 0;

cout<<"Insert the element in queue : "<<endl;

cin>>val;

rear++;

queue[rear] = val;

}

}

void dequeue() {

if (front == - 1 || front > rear) {

cout<<"Queue Underflow ";

return ;

} else {

cout<<"Element deleted from queue is : "<< queue[front] <<endl;

front++;;

}

}

void display() {

if (front == - 1)

cout<<"Queue is empty"<<endl;

else {

cout<<"Queue elements are : ";

for (int i = front; i <= rear; i++)

cout<<queue[i]<<" ";

cout<<endl;

}

}

};

int queue::n=20;

int queue::rear=-1;

int queue::front=-1;

class array

{

int a[10];

int n;

public:

void insert(int n)

{

cout<<"enter element to insert:";

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

cin>>a[i];

}

int del(int x,int n)

{

int i;

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

if (a[i] == x)

break;

if (i < n)

{

n = n - 1;

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

a[j] = a[j+1];

}

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

cout<<a[i]<<endl;

}

void display(int n)

{

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

cout<<a[i]<<endl;

}

};

int main()

{

Linkedlist list;

list.insertNode(1);

list.insertNode(2);

list.insertNode(3);

list.insertNode(4);

cout << "Elements of the list are: ";

list.printList();

cout << endl;

list.deleteNode(3);

cout << "Elements of the list are: ";

list.printList();

cout << endl;

stack s;

int ch, val;

cout<<"1) Push in stack"<<endl;

cout<<"2) Pop from stack"<<endl;

cout<<"3) Display stack"<<endl;

cout<<"4) Exit"<<endl;

do {

cout<<"Enter choice: "<<endl;

cin>>ch;

switch(ch) {

case 1: {

cout<<"Enter value to be pushed:"<<endl;

cin>>val;

s.push(val);

break;

}

case 2: {

s.pop();

break;

}

case 3: {

s.display();

break;

}

case 4: {

cout<<"Exit"<<endl;

break;

}

default: {

cout<<"Invalid Choice"<<endl;

}

}

}while(ch != 4);

queue q;

int c;

cout<<"1) Insert element to queue"<<endl;

cout<<"2) Delete element from queue"<<endl;

cout<<"3) Display all the elements of queue"<<endl;

cout<<"4) Exit"<<endl;

do {

cout<<"Enter your choice : "<<endl;

cin>>c;

switch (c) {

case 1: q.enqueue();

break;

case 2: q.dequeue();

break;

case 3: q.display();

break;

case 4: cout<<"Exit"<<endl;

break;

default: cout<<"Invalid choice"<<endl;

}

} while(c!=4);

array a;

int n;

cout<<"enter size of array:";

cin>>n;

a.insert(n);

a.display(n);

int x;

cout<<"enter value to be deleted:";

cin>>x;

a.del(x,n);

return 0;

}

**Output:**

Enter value to be pushed:

3

Enter choice:

3

Stack elements are:3

Enter choice:

2

The popped element is 3

Enter choice:

4

Exit

1) Insert element to queue

2) Delete element from queue

3) Display all the elements of queue

4) Exit

Enter your choice :

1

Insert the element in queue :

5

Enter your choice :

1

Insert the element in queue :

5

Enter your choice :

3

Queue elements are : 5 5

Enter your choice :

2

Element deleted from queue is : 5

Enter your choice :

5

Invalid choice

Enter your choice :

4

Exit

enter size of array:3

enter element to insert:1

1

1

1

1

1

enter value to be deleted:1

1

1

Press any key to continue . . .1