|  |  |
| --- | --- |
| DS Lab practical file  Submitted to: Mrs. Anjali | Abstract  Source code and other files can be found at: https://github.com/Avchhikara/ds\_lab  Avnish    Roll no: 17001003017  Branch: Btech(ECE) - 1st Year |
|  |  |

Index

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Experiment** | **Date** | **Signature** |
| **1.**  **2.**  **3.**  **4.**  **5.**  **6.**  **7.**  **8.**  **9.**  **10.** | **To perform binary search in an array.**  **To perform various operation on matrices.**  **To swap two numbers using call by reference and call by value.**  **To implement bubble sort.**  **To implement insertion sort.**  **To implement selection sort.**  **Link list implementation of Stack.**  **Link list implementation of Queue.**  **To search an element in a link list.**  **To maintain a link list.** |  |  |

**Experiment - 1**

**Write a program to perform binary search in an array.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

//To generate a random array

void create\_array(int \*arr, int len){

int i = 0;

while(i<len)

arr[i++] = rand()%len;

}

void print\_array(int \*arr, int len){

int i = 0;

while(i<len)

cout<<arr[i++]<<" ";

}

//Now, sorting using insertion sort

void sort\_array(int \*arr, int len){

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

int key = arr[i];

int j = i-1;

while(j>=0 && key<arr[j]){

arr[j+1] = arr[j];

j = j-1;

}

arr[j+1] = key;

}

}

int binary\_search(int \*arr, int len, int term){

int beg = 0, end = len-1;

while(beg+1 < end){

int mid = (beg+end)/2;

if(term==arr[mid])

return mid;

else if(term>arr[mid])

beg = mid;

else if(term<arr[mid])

end = mid;

}

return -1;

}

int main(){

int len, term = -1;

cout<<"\nEnter the length of array to be generated: ";

cin>>len; //Getting the length of array

int \*arr = (int \*)malloc(sizeof(int)\*len); //allocating memory to integer pointer

create\_array(arr, len); //Create a random array of specified length

cout<<"\nGenerated array in sorted from is: ";

sort\_array(arr, len); //Sorting the array

print\_array(arr, len); //Printing for user's reference

//Now, getting the keyword to be searched

cout<<"\n Enter the term to be searched: ";

cin>>term;

int out = binary\_search(arr, len, term);

if(out==-1)

cout<<"\n NOT FOUND ";

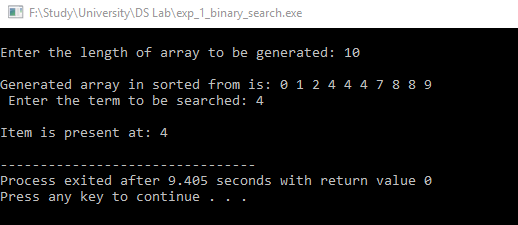
else

cout<<"\nItem is present at: "<<out<<endl;

return 0;

}

**Output:**

****

**Experiment - 2**

**Write a program to perform various operation on matrices.**

**Solution:**

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

using namespace std;

void add(int \*arr\_1, int \*arr\_2, int \*out){

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

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

\*(out + i + j) = \*(arr\_1 + i + j) + \*(arr\_2 + i + j);

}

}

}

void sub(int \*arr\_1, int \*arr\_2, int \*out){

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

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

\*(out + i + j) = \*(arr\_1 + i + j) - \*(arr\_2 + i + j);

}

}

}

void mul(int \*arr\_1, int \*arr\_2, int \*out){

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

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

\*(out + i +j) = 0;

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

\*(out + i +j) = (\*(arr\_1 + i + k)) \* (\*(arr\_2 + k + j));

}

}

}

}

void print\_out(int \*arr\_1, int \*arr\_2, int \*out, char c){

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

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

printf("%3d", \*(arr\_1 + i + j));

}

if(i == 1){

printf(" %c ", c);

}

else

printf(" ");

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

printf("%3d", \*(arr\_2 + i+ j));

printf("%3s", i==1? " = ": " ");

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

printf("%7d", \*(out + i + j));

cout<<endl;

}

}

int main(){

cout<<"\n The operations you can perform are: \n1. Enter 1 to add the given arrays\n2. Enter 2 to subtract two arrays\n3. Enter 3 to multiply two arrays\n4. Enter -1 to exit the program\n";

int arr\_1[3][3], arr\_2[3][3], out[3][3];

//Generating the two array

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

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

arr\_1[i][j] = rand()%100;

arr\_2[i][j] = rand()%100;

}

}

//Printing the array

cout<<"\nThe arrays we have are: \n";

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

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

printf( "%2d " , arr\_1[i][j]);

}

printf("%s", i==1?" And ": " ");

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

printf( "%2d " , arr\_2[i][j]);

cout<<endl;

}

while(true){

int inp=0;

cout<<"\nEnter the operation you want to perform: ";

cin>>inp;

if(inp == 1){

add(arr\_1[0], arr\_2[0], out[0]);

print\_out(arr\_1[0], arr\_2[0], out[0], '+');

}

else if(inp == 2){

sub(arr\_1[0], arr\_2[0], out[0]);

print\_out(arr\_1[0], arr\_2[0], out[0], '-');

}

else if(inp == 3){

mul(arr\_1[0], arr\_2[0], out[0]);

print\_out(arr\_1[0], arr\_2[0], out[0], '\*');

}

else{

cout<<"\n The program will exit now!! \n";

break;

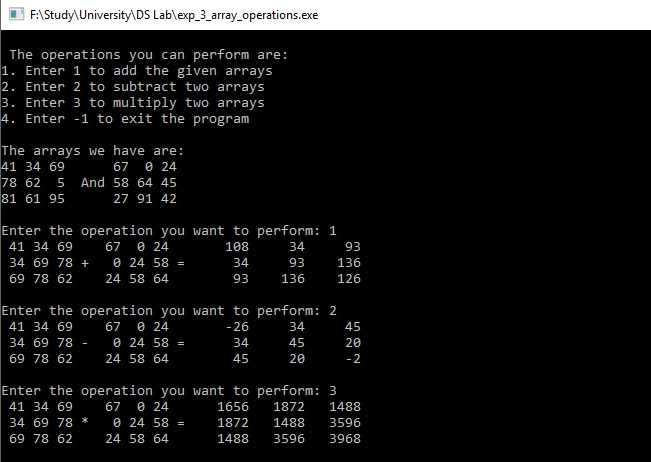
}

}

return 0;

}

**Output:**

****

**Experiment - 3**

**Write a program to swap two nos. using calls by value and reference.**

**Solution:**

#include <iostream>

using namespace std;

void call\_by\_reference(int \*a, int \*b){

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void call\_by\_value(int a, int b){

int temp = a;

a = b;

b = temp;

cout<<"a="<<a<<" and b="<<b;

}

int main(){

int a, b;

cout<<"\nEnter numbers to be swapped: ";

cin>>a>>b;

cout<<"\nSwapped output----> ";

cout<<"\tCall by value: ";

call\_by\_value(a, b);

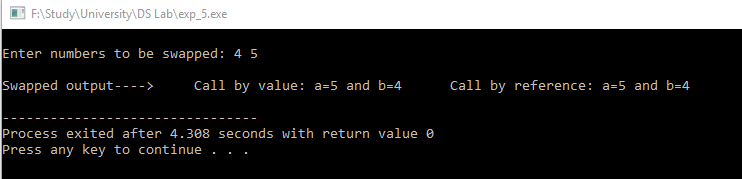
call\_by\_reference(&a, &b);

cout<<"\tCall by reference: a="<<a<<" and b="<<b<<endl;

return 0;

}

**Output:**

****

**Experiment - 4**

**Write a program to implement bubble sort.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

//To generate a random array

void create\_array(int \*arr, int len){

int i = 0;

while(i<len)

arr[i++] = rand()%len;

}

void print\_array(int \*arr, int len){

int i = 0;

while(i<len)

cout<<arr[i++]<<" ";

}

//for swapping two numbers

void swap(int \*a, int \*b){

int temp = \*a;

\*a = \*b;

\*b = temp;

}

//implementation of bubble sort

void bubble\_sort(int \*arr, int len){

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

int var = arr[i];

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

if(arr[j]<arr[i])

swap(&arr[j], &arr[i]);

}

}

int main(){

int len = 10;

int \*arr = (int \*)malloc(sizeof(int )\*len);

create\_array(arr, len);

cout<<"\nUnsorted array is: ";

print\_array(arr, len);

bubble\_sort(arr, len);

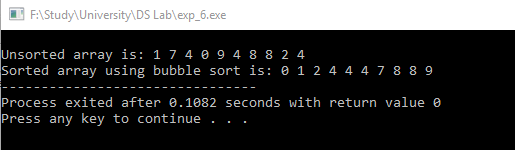
cout<<"\nSorted array using bubble sort is: ";

print\_array(arr, len);

return 0;

}

**Output:**

****

**Experiment - 5**

**Write a program to implement insertion sort.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

//To generate a random array

void create\_array(int \*arr, int len){

int i = 0;

while(i<len)

arr[i++] = rand()%len;

}

void print\_array(int \*arr, int len){

int i = 0;

while(i<len)

cout<<arr[i++]<<" ";

}

//Now, sorting using insertion sort

void insertion\_sort(int \*arr, int len){

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

int key = arr[i];

int j = i-1;

while(j>=0 && key<arr[j]){

arr[j+1] = arr[j];

j = j-1;

}

arr[j+1] = key;

}

}

int main(){

int len = 10;

int \*arr = (int \*)malloc(sizeof(int )\*len);

create\_array(arr, len);

cout<<"\nUnsorted array: ";

print\_array(arr, len);

cout<<"\nSorted array with insertion sort is: ";

insertion\_sort(arr, len);

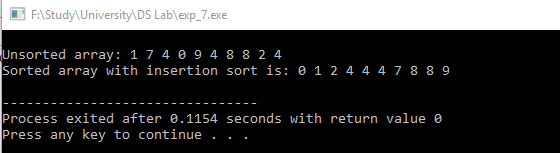
print\_array(arr, len);

cout<<endl;

return 0;

}

**Output:**

****

**Experiment - 6**

**Write a program to implement selection sort.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

//To generate a random array

void create\_array(int \*arr, int len){

int i = 0;

while(i<len)

arr[i++] = rand()%len;

}

void print\_array(int \*arr, int len){

int i = 0;

while(i<len)

cout<<arr[i++]<<" ";

}

void swap(int \*a, int \*b){ //for swapping two numbers

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void selection\_sort(int \*arr, int len) {

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

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

{

// Find the minimum element in unsorted array

min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

swap(&arr[min\_idx], &arr[i]);

}

}

int main(){

int len = 10;

int \*arr = (int \*)malloc(sizeof(int )\*len);

create\_array(arr, len);

cout<<" Unsorted array is: ";

print\_array(arr, len);

cout<<"\nSorted array(using selection sort) is: ";

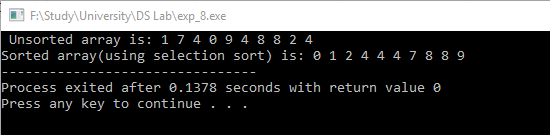
selection\_sort(arr, len);

print\_array(arr, len);

return 0;

}

**Output:**

****

**Experiment - 7**

**Write a program of link list implementation of a stack.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

struct list\_node{

int value;

struct list\_node \*next;

};

typedef struct list\_node list\_node;

struct stack{

int size;

list\_node \*top;

};

typedef struct stack stack;

stack \*create\_stack(){

stack \*s = (stack \*)malloc(sizeof(stack));

s->size = 0;

s->top = NULL;

return s;

}

void push(stack \*s , int value){

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node));

ln->value = value;

ln->next = NULL;

s->size++;

if(s->top==NULL){

s->top = ln;

}

else{

ln->next = s->top;

s->top = ln;

}

}

int pop(stack \*s){

int value = s->top->value;

list\_node \*pop = s->top;

s->top = s->top->next;

pop->next = NULL;

return value;

}

int main(){

stack \*s = create\_stack();

cout<<"Enter the value to be pushed to stack or -1 to stop pushing: ";

int inp;

cin>>inp;

while(inp!=-1){

push(s, inp);

cin>>inp;

}

cout<<"\n Stack you have is: ";

while(s->top!=NULL){

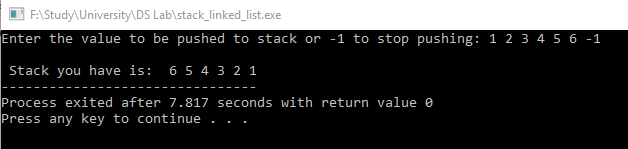
cout<<" "<<pop(s);

}

return 0;

}

**Output:**

****

**Experiment - 8**

**Write a program of linked list implementation of a queue.**

**Solution:**

//Linked list implementation of a queue

//Linked list implementation of a queue

#include <iostream>

#include <stdlib.h>

using namespace std;

struct list\_node {

int value;

struct list\_node \*next;

};

typedef struct list\_node list\_node;

struct queue{

int size;

list\_node \*front;

list\_node \*back;

};

typedef struct queue queue;

queue \*create\_queue(){

queue \*q = (queue \*)malloc(sizeof(queue));

q->size = 0;

q->front = NULL;

q->back = NULL;

return q;

}

void enqueue(queue \*q, int value){

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node ));

ln->value = value;

if(q->size==0){

q->front = ln;

q->back = ln;

q->size++;

}

else{

q->back->next = ln;

q->back = ln;

q->size++;

}

}

int dequeue(queue \*q){

if(q->size==0){

return -1;

}

list\_node \*dequeue = q->front;

int value = dequeue->value;

q->front = q->front->next;

q->size--;

dequeue->next = NULL; //Freeing up the dequeued node

return value;

}

int main(){

queue \*q = create\_queue();

cout<<"Enter the value to be enqueued to Queue or -1 to stop enquiuing: ";

int inp;

cin>>inp;

while(inp!=-1){

enqueue(q, inp);

cin>>inp;

}

cout<<"\n Queue you have is: ";

while(q->size!=0){

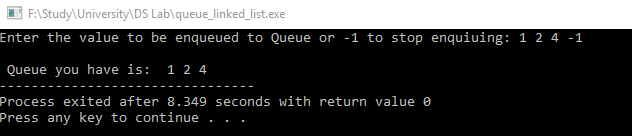
cout<<" "<<dequeue(q);

}

return 0;

}

**Output:**

****

**Experiment - 9**

**Write a program to search an element in a link list.**

**Solution:**

#include <iostream>

#include <stdlib.h>

using namespace std;

struct list\_node{

int value;

struct list\_node \*next;

};

typedef struct list\_node list\_node;

struct list{

int size;

list\_node \*head;

};

typedef struct list list;

list \*create\_linked\_list(){

list \*l = (list \*)malloc(sizeof(list));

l->size = 0;

l->head = NULL;

return l;

}

void push(list \*l, int value){

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node));

ln->value = value;

ln->next = NULL;

if(l->size==0){

l->head = ln;

}

else{

ln->next = l->head;

l->head = ln;

}

l->size++;

}

int pop(list \*l){

if(l->size==0){

return -1;

}

list\_node \*pop = l->head;

int value = pop->value;

l->head = l->head->next;

pop->next = NULL; //Freeing up the popped out node

l->size--;

return value;

}

void view(list \*l){

cout<<"\n The linked list we have is: ";

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node));

ln = l->head;

for(int i=0; i<l->size; i++){

cout<<" "<<ln->value;

ln = ln->next;

}

}

void search\_list(list \*l, int inp){

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node ));

ln = l->head;

for(int i=0; i<l->size; i++){

if(ln->value==inp){

cout<<"It's present at : "<<i;

return;

}

ln = ln->next;

}

cout<<"\n\tValue not present\n";

}

int main(){

list \*l = create\_linked\_list();

int inp;

cout<<"\nWelcome to linked list maintainance program. Here are your options -->";

cout<<"\n1. Enter 1 for adding enteries to the list.";

cout<<"\n2. Enter 2 for viewing the linked list.";

cout<<"\n3. Enter 3 to search for a item.";

cout<<"\n4. Enter 4 for deleting the entire list.";

cout<<"\nENTER -1 TO EXIT PROGRAM";

while(true){

cout<<"\nEnter your input: ";

cin>>inp;

if(inp==1){

cout<<"\n Enter the value you want to add: ";

cin>>inp;

push(l, inp);

cout<<"\nValue added";

}

else if(inp==2){

view(l);

}

else if(inp==3){

view(l);

cout<<"\n Enter the value, to search for: ";

cin>>inp;

search\_list(l, inp);

}

else if(inp==4){

while(l->size!=0){

pop(l);

}

cout<<"\n\tWhole list has been deleted\n";

}

else{

cout<<"\nThe program will terminate now\n";

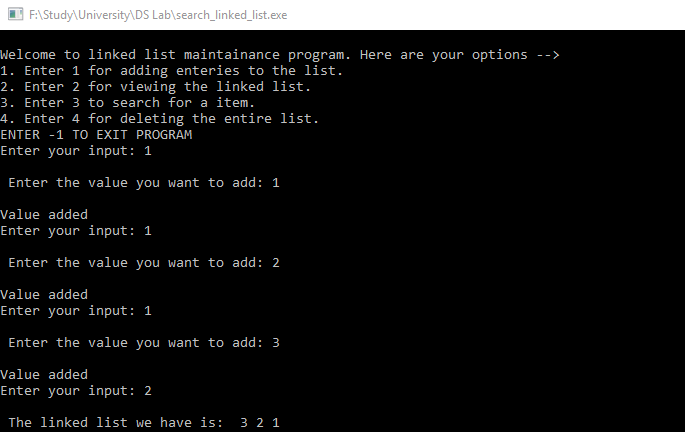
break;

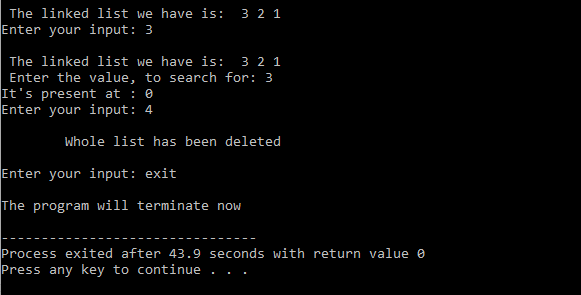
}

}

return 0;

}

**Output:**

****

**Experiment - 10**

**Write a program to maintain a linked list.**

**Solution:**

//Linked list implementation

#include <iostream>

#include <stdlib.h>

using namespace std;

struct list\_node{

int value;

struct list\_node \*next;

};

typedef struct list\_node list\_node;

struct list{

int size;

list\_node \*head;

};

typedef struct list list;

list \*create\_linked\_list(){

list \*l = (list \*)malloc(sizeof(list));

l->size = 0;

l->head = NULL;

return l;

}

void push(list \*l, int value){

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node));

ln->value = value;

ln->next = NULL;

if(l->size==0){

l->head = ln;

}

else{

ln->next = l->head;

l->head = ln;

}

l->size++;

}

int pop(list \*l){

if(l->size==0){

return -1;

}

list\_node \*pop = l->head;

int value = pop->value;

l->head = l->head->next;

pop->next = NULL; //Freeing up the popped out node

l->size--;

return value;

}

void view(list \*l){

cout<<"\n The linked list we have is: ";

list\_node \*ln = (list\_node \*)malloc(sizeof(list\_node));

ln = l->head;

for(int i=0; i<l->size; i++){

cout<<" "<<ln->value;

ln = ln->next;

}

}

int main(){

list \*l = create\_linked\_list();

int inp;

cout<<"\nWelcome to linked list maintainance program. Here are your options -->";

cout<<"\n1. Enter 1 for adding enteries to the list.";

cout<<"\n2. Enter 2 for viewing the linked list.";

cout<<"\n3. Enter 3 for deleting the entire list.";

cout<<"\nENTER -1 TO EXIT PROGRAM";

while(true){

cout<<"\nEnter your input: ";

cin>>inp;

if(inp==1){

cout<<"\n Enter the value you want to add: ";

cin>>inp;

push(l, inp);

cout<<"\nValue added";

}

else if(inp==2){

view(l);

}

else if(inp==3){

while(l->size!=0){

pop(l);

}

}

else{

cout<<"\nThe program will terminate now\n";

break;

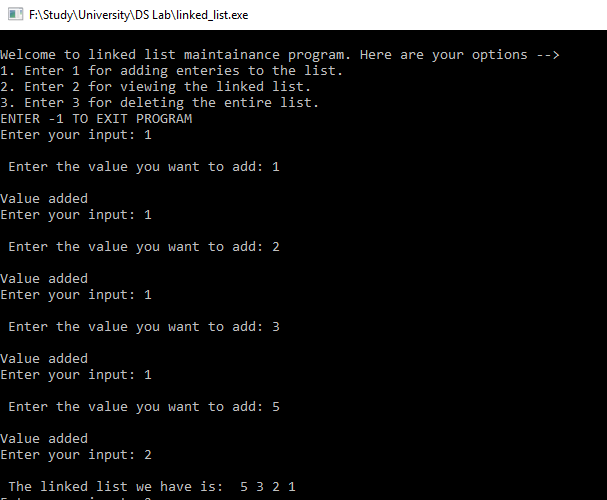
}

}

return 0;

}

**Output:**

****