---------------------------17 –02 – 2021 ------------------------

// Stack example without pointers

#include <stdio.h>

int main ()

{

//int stack[5]={1,2,3,4}; // Cannot be done like this

int stack[5]; int top=-1; int choice=0;

while(choice != 4)

{

menu();

scanf("%d")

switch(choice)

{

case 1:

case 2:

case 3:

case 4: choice = 4; default;

}

}

return 0;

}

void menu()

{

printf("Enter your choice\n");

printf("1.Push \n 2. Pop \n 3. Print \n 4.Exit \n");

}

push(){}

pop(){}

----------------------------stack without print -------------------

// Stack example without pointers

#include <stdio.h>

int main ()

{

//int stack[5]={1,2,3,4}; // Cannot be done like this

int stack[5], top=-1,choice=0,size=5;

while(choice != 4)

{

menu();

scanf("%d")

switch(choice)

{

case 1:

if(top == size - 1)

printf("\n Stack Full Cannot Push ");

break;

else

push();

break;

case 2: if(top == - 1)

printf("\n Stack is Empty Cannot Pop \n ");

break;

else

pop();

break;

case 3:

case 4: choice = 4; default;

}

}

return 0;

}

void menu()

{

printf("Enter your choice\n");

printf("1.Push \n 2. Pop \n 3. Print \n 4.Exit \n");

return;

}

void push()

{

printf("Enter element to push");

scanf("%d",&element);

++top;

stack[top]=element;

printf("Pushed %d",element);

return;

}

void pop()

{

printf("Poped Element is %d ",stack[top]);

--top;

return;

}

void print(){

}

----------------------- Full Stack Program ---------------------

// Stack example without pointers

#include <stdio.h>

int stack[5], top=-1,choice=0,size=5,element;

int main ()

{

//int stack[5]={1,2,3,4}; // Cannot be done like this

while(choice != 4)

{

menu();

scanf("%d",&choice);

switch(choice)

{

case 1:

if(top == size - 1)

{

printf("\n Stack Full Cannot Push ");

break;

}

else

{

push();

}

break;

case 2: if(top == - 1)

{printf("\n Stack is Empty Cannot Pop \n ");

break;}

else

{pop();}

break;

case 3: print(); break;

case 4: choice = 4; break;

default: printf("\n Wrong choice");

break;

}

}

return 0;

}

void menu()

{

printf("Enter your choice\n");

printf("1.Push \n 2. Pop \n 3. Print \n 4.Exit \n");

return;

}

void push()

{

printf("Enter element to push");

scanf("%d",&element);

++top;

stack[top]=element;

printf("Pushed %d",element);

return;

}

void pop()

{

printf("Poped Element is %d ",stack[top]);

--top;

return;

}

void print()

{

printf("-------Stack is ------- \n");

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

printf("\n %d \n",stack[i]);

}

}

--------------------------------- Linked List example without pointers without loop -------

// Linked List example without pointers without loop

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

int main ()

{

ll l1,l2,l3,l4;

l1.data=2;

l2.data=4;

l3.data=6;

l4.data=8;

l1.link = &l2 ;

l2.link = &l3;

l3.link = &l4;

l4.link = NULL;

printf("\n %d %u --> ",l1.data,l1.link);

printf(" %d %u --> ",l2.data,l2.link);

printf(" %d %u --> ",l3.data,l3.link);

printf(" %d %u --> \n",l4.data,l4.link);

return 0;

}

--------------------------Linked List with dynamic memory allocation --------

----------insert beg and print working -----------

// Linked List example with dynamic memory allocation without loop

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

ll \*first=NULL,\*temp ;int data=0,choice=0;

int main ()

{

while(choice != 4)

{

menu();

scanf("%d",&choice);

switch(choice)

{

case 1:

break;

case 2:

insertbeg();

break;

case 3:break;

case 4:break;

case 5:break;

case 6:break;

case 7:print(); break;

case 8:break;

case 9: exit(0);

default : printf("Wrong choice");

}

}

return 0;

}

void menu()

{

printf("\nEnter your choice\n");

printf("1.Insert Beg \n2.Insert End \n3.Insert at position \n");

printf("4.Del First \n5.Delete Last \n6.Delete node at position \n");

printf("7.Print \n8.Search \n9.Exit \n");

return;

}

void insertbeg()

{

if(first == NULL) // if the list is empty

{

first = (ll \*)malloc(sizeof(ll));

printf("Enter data for the node \n");

scanf("%d",&data);

first->data = data;

first->link = NULL;

}

else

{

temp = (ll \*)malloc(sizeof(ll));

printf("Enter data for the node \n");

scanf("%d",&data);

temp->data = data;

temp->link = first;

first = temp;

}

}

void insertend()

{

}

void insertatpos()

{

}

void deleteatbeg()

{

}

void deletelast()

{

}

void deletepos()

{

}

void print(){

if(first == NULL)

{

printf("List is empty ");

}

else

{

temp = first;

while(temp != NULL)

{

printf(" [%d %u]-> \t",temp->data,temp->link);

temp = temp -> link;

}

}

}

void search(){

}

-------------------------18-2-2021 -----------------------------

//----------insert beg and print working and insert after position -----------

// Linked List example with dynamic memory allocation without loop

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

ll \*first=NULL,\*temp ;int data=0,choice=0,sizeoflist;

int main ()

{

while(choice != 4)

{

menu();

scanf("%d",&choice);

switch(choice)

{

case 1:

break;

case 2:

insertbeg();

break;

case 3:

insertatpos();

break;

case 4:break;

case 5:break;

case 6:break;

case 7:print(); break;

case 8:break;

case 9: exit(0);

default : printf("Wrong choice");

}

}

return 0;

}

void menu()

{

printf("\nEnter your choice\n");

printf("1.Insert End \n2.Insert Beg \n3.Insert after which position \n");

printf("4.Del First \n5.Delete Last \n6.Delete node at position \n");

printf("7.Print \n8.Search \n9.Exit \n");

return;

}

void insertbeg()

{

if(first == NULL) // if the list is empty

{

first = (ll \*)malloc(sizeof(ll));

++sizeoflist;

printf("Enter data for the node \n");

scanf("%d",&data);

first->data = data;

first->link = NULL;

}

else

{

temp = (ll \*)malloc(sizeof(ll));

printf("Enter data for the node \n");

scanf("%d",&data);

temp->data = data;

temp->link = first;

first = temp;

}

}

void insertend()

{

}

void insertatpos()

{

int position=0; int tdata; ll \* temp1;

printf("Enter after which node you want to insert node ");

scanf("%d",&position);

if(first == NULL)

{

printf("Linked List Does not exit ");

return;

}

if(position == 0)

{

insertbeg();

return;

}

if(position > sizeoflist )

{

printf("Out of bounds ");

return ;

}//

// 3

temp = first;

for (int i = 1; i < position; i++) {

temp = temp->link;

}

printf("Enter data");

scanf("%d",&tdata);

temp1 = (ll \*) malloc(sizeof(ll));

temp1->link = temp->link;

temp->link = temp1;

temp1->data = tdata;

++sizeoflist;

}

void deleteatbeg()

{

}

void deletelast()

{

}

void deletepos()

{

}

void print(){

if(first == NULL)

{

printf("List is empty ");

}

else

{

temp = first;

while(temp != NULL)

{

printf(" [%d %u]-> \t",temp->data,temp->link);

temp = temp -> link;

}

}

}

void search(){

}

----------------------------------- except search ---------------------

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

ll \*first=NULL,\*temp ;int data=0,choice=0,sizeoflist;

int main ()

{

while(choice != 9)

{

menu();

scanf("%d",&choice);

switch(choice)

{

case 1:

break;

case 2:

insertbeg();

break;

case 3:

insertatpos();

break;

case 4:

deleteatbeg();

break;

case 5: deletepos();

break;

case 6:break;

case 7:print(); break;

case 8:break;

case 9: exit(0);

default : printf("Wrong choice");

}

}

return 0;

}

void menu()

{

printf("\nEnter your choice\n");

printf("1.Insert End \n2.Insert Beg \n3.Insert after which position \n");

printf("4.Del First \n5.Delete Node no \n");

printf("7.Print \n8.Search \n9.Exit \n");

return;

}

void insertbeg()

{

if(first == NULL) // if the list is empty

{

first = (ll \*)malloc(sizeof(ll));

++sizeoflist;

printf("Enter data for the node \n");

scanf("%d",&data);

first->data = data;

first->link = NULL;

}

else

{

temp = (ll \*)malloc(sizeof(ll));

printf("Enter data for the node \n");

scanf("%d",&data);

temp->data = data;

temp->link = first;

first = temp;

}

}

void insertend()

{

}

void insertatpos()

{

int position=0; int tdata; ll \* temp1;

printf("Enter after which node you want to insert node ");

scanf("%d",&position);

if(first == NULL)

{

printf("Linked List Does not exit ");

return;

}

if(position == 0)

{

insertbeg();

return;

}

if(position > sizeoflist )

{

printf("Out of bounds ");

return ;

}//

// 3

temp = first;

for (int i = 1; i < position; i++) {

temp = temp->link;

}

printf("Enter data");

scanf("%d",&tdata);

temp1 = (ll \*) malloc(sizeof(ll));

temp1->link = temp->link;

temp->link = temp1;

temp1->data = tdata;

++sizeoflist;

}

void deleteatbeg()

{

ll \* temp1;

if(first == NULL){

printf("No List ");

return ;

}

printf("Deleted Node Data is %d",first->data);

temp1 = first;

first= first->link;

free(temp1);

}

void deletelast()

{

}

void deletepos()

{

ll \* temp,\*temp1; int pos;

temp = first;

printf("Enter Position to delete ");

scanf("%d",&pos);

if(pos == 0)

{

deleteatbeg();

return;

}

for (int i = 1; i < pos; i++) {

temp1 = temp;

temp = temp -> link;

}

temp1->link = temp->link;

printf("Deleted Data is %d \n",temp->data);

free(temp);

}

void print(){

if(first == NULL)

{

printf("List is empty ");

}

else

{

temp = first;

while(temp != NULL)

{

printf(" [%d %u]-> \t",temp->data,temp->link);

temp = temp -> link;

}

}

}

void search(){

}

----------------------Linked List C,R,U,D and Search -------------

//----------insert beg and print working -----------

// Linked List example with dynamic memory allocation without loop

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

ll \*first=NULL,\*temp ;int data=0,choice=0,sizeoflist;

int main ()

{

while(choice != 9)

{

menu();

scanf("%d",&choice);

switch(choice)

{

case 1:

break;

case 2:

insertbeg();

break;

case 3:

insertatpos();

break;

case 4:

deleteatbeg();

break;

case 5: deletepos();

break;

case 6:break;

case 7:print(); break;

case 8:

search();

break;

case 9: exit(0);

default : printf("Wrong choice");

}

}

return 0;

}

void menu()

{

printf("\nEnter your choice\n");

printf("1.Insert End \n2.Insert Beg \n3.Insert after which position \n");

printf("4.Del First \n5.Delete Node no \n");

printf("7.Print \n8.Search \n9.Exit \n");

return;

}

void insertbeg()

{

if(first == NULL) // if the list is empty

{

first = (ll \*)malloc(sizeof(ll));

++sizeoflist;

printf("Enter data for the node \n");

scanf("%d",&data);

first->data = data;

first->link = NULL;

}

else

{

temp = (ll \*)malloc(sizeof(ll));

printf("Enter data for the node \n");

scanf("%d",&data);

temp->data = data;

temp->link = first;

first = temp;

}

}

void insertend()

{

}

void insertatpos()

{

int position=0; int tdata; ll \* temp1;

printf("Enter after which node you want to insert node ");

scanf("%d",&position);

if(first == NULL)

{

printf("Linked List Does not exit ");

return;

}

if(position == 0)

{

insertbeg();

return;

}

if(position > sizeoflist )

{

printf("Out of bounds ");

return ;

}//

// 3

temp = first;

for (int i = 1; i < position; i++) {

temp = temp->link;

}

printf("Enter data");

scanf("%d",&tdata);

temp1 = (ll \*) malloc(sizeof(ll));

temp1->link = temp->link;

temp->link = temp1;

temp1->data = tdata;

++sizeoflist;

}

void deleteatbeg()

{

ll \* temp1;

if(first == NULL){

printf("No List ");

return ;

}

printf("Deleted Node Data is %d",first->data);

temp1 = first;

first= first->link;

free(temp1);

}

void deletelast()

{

}

void deletepos()

{

ll \* temp,\*temp1; int pos;

temp = first;

printf("Enter Position to delete ");

scanf("%d",&pos);

if(pos == 0)

{

deleteatbeg();

return;

}

for (int i = 1; i < pos; i++) {

temp1 = temp;

temp = temp -> link;

}

temp1->link = temp->link;

printf("Deleted Data is %d \n",temp->data);

free(temp);

}

void print(){

if(first == NULL)

{

printf("List is empty ");

}

else

{

temp = first;

while(temp != NULL)

{

printf(" [%d %u]-> \t",temp->data,temp->link);

temp = temp -> link;

}

}

}

void search(){

int searchelement=0,foundindex=0,searchposition,flag=0;

ll \* searchpointer,\*stemp;

if(first == NULL)

{

printf("List Empty");

return;

}

stemp = first;

printf("Enter element to search ");

scanf("%d",&searchelement);

for (int i = 1; stemp != NULL ; i++) {

if((stemp->data) == searchelement)

{

flag = 1 ;

Searchpointer = stemp;

searchposition = i;

break;

}

stemp=stemp->link;

}

if(flag == 1)

{

printf("Found %d at %d \n",searchpointer->data,searchposition);

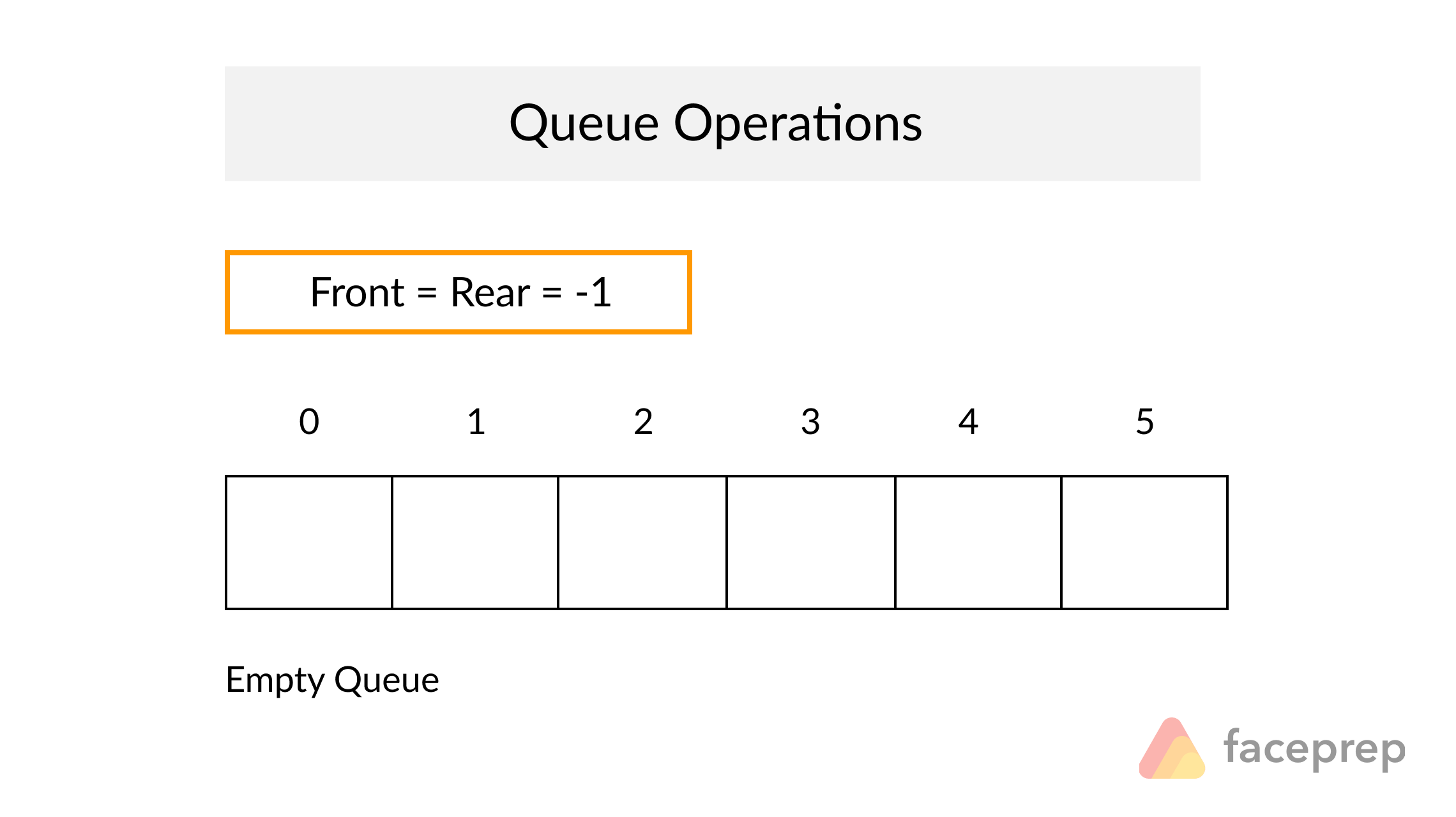
}

else

printf("%d not found",searchelement);

}

-------------------------------- Queue ------------



#include <stdio.h>

int f=0,r=-1,size=6;int ch=0;int data=0;

int queue[6];

int

main ()

{

int queue[6];

while(ch!=4)

{

printf("\n1.enqueue \n 2.dequeue \n 3.print \n4.Exit \n");

printf("Enter your choice ");

scanf("%d",&ch);

switch(ch)

{

case 1: enqueue();break;

case 2: dequeue(); break;

case 3: print(); break;

case 4: exit(0);

}

}

}

void enqueue()

{

if((r==size-1))

{

printf("queue full");

return;

}

++r;

printf("Enter Data to enqueue");

scanf("%d",&data);

queue[r]=data;

printf("Queued Data\n");

}

void dequeue(){

if((f==0) && (r==-1))

{

printf("Q is empty ");

return;

}

printf("dequeued data is %d",queue[f]);

++f;

}

void print(){

for (int i = f; i <= r; i++) {

printf("%d \t ",queue[i]);

/\* code \*/

}

}

------------------19-02-2021 ---------------------

-----------------------------Linked Stack ---------------

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

ll \* first = NULL, \*top=NULL;

int data = 0, choice = 0, sizeoflist;

int

main ()

{

while (choice != 4)

{

menu ();

scanf ("%d", &choice);

switch (choice)

{

case 1: push(); break;

case 2: pop(); break ;

case 3: print(); break;

case 4: exit (0);

default: printf ("Wrong choice");

}

}

return

0;

}

void

menu (){

printf("--------Linked Stack -----");

printf ("\nEnter your choice \n");

printf

(" 1.Push \n 2.Pop \n 3.Print \n 4.Exit \n");

return;

}

void push(){

ll \*temp;int i;ll \* temp1;

// Gets executd only Once

if(first == NULL) // When the stack is empty

{

first = (ll\*) malloc(sizeof(ll));

printf("Enter Data ");

scanf("%d",&data);

first->data =data;

first->link = NULL;

top = first;

return;

}

temp = first;

for(i=0;temp->link != NULL;)

{

temp=temp->link;

}

temp1 = (ll\*) malloc(sizeof(ll));

printf("Enter Data ");

scanf("%d",&data);

temp1->data =data;

temp->link = temp1;

return;

}

void pop()

{

ll \*t1=NULL, \*t2=NULL;

if(first == NULL)

{

printf("Stack Empty ");

return;

}

t2 = first;

for(;t2->link != NULL;)

{

t1 = t2;

t2=t2->link;

}

printf("Poped Element is %d \n",t2->data);

t1->link = NULL;

free(t2);

}

void print(){

ll \*t2=NULL;

if(first == NULL)

{

printf("Stack Empty ");

return;

}

t2 = first;

for(;t2!=NULL;)

{printf("%d \t",t2->data);

t2=t2->link;

}

}

--------------------------------------Lined Queue --------------------

// linked queue example with linked list

#include <stdio.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

ll \* first = NULL, \*top=NULL;

int data = 0, choice = 0, sizeoflist;

int

main ()

{

while (choice != 4)

{

menu ();

scanf ("%d", &choice);

switch (choice)

{

case 1: enqueue(); break;

case 2: dequeue(); break ;

case 3: print(); break;

case 4: exit (0);

default: printf ("Wrong choice");

}

}

return

0;

}

void

menu (){

printf("--------Linked Stack -----");

printf ("\nEnter your choice \n");

printf

("\n1.Enqueue \n2.Dqueue \n3.Print \n4.Exit \n");

return;

}

void enqueue(){

ll \*temp;int i;ll \* temp1;

// Gets executd only Once

if(first == NULL) // When the queue is empty

{

first = (ll\*) malloc(sizeof(ll));

printf("Enter Data ");

scanf("%d",&data);

first->data =data;

first->link = NULL;

top = first;

return;

}

temp = first;

for(i=0;temp->link != NULL;)

{

temp=temp->link;

}

temp1 = (ll\*) malloc(sizeof(ll));

printf("Enter Data ");

scanf("%d",&data);

temp1->data =data;

temp->link = temp1;

return;

}

void dequeue() // delete first of Linked list is equal to dequeue

{

ll \*t1=NULL, \*t2=NULL;

if(first == NULL)

{

printf("Stack Empty ");

return;

}

t2 = first;

printf("Dequeued %d \n",first->data);

first = first = first -> link;

free(t2);

}

void print(){

ll \*t2=NULL;

if(first == NULL)

{

printf("Stack Empty ");

return;

}

t2 = first;

for(;t2!=NULL;)

{printf("%d \t",t2->data);

t2=t2->link;

}

}

----------------------------22-02-2021 ----------------------

Command line args example

#include <stdio.h>

int main( int argc, char \*argv[] ) {

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

{

printf("%s",argv[i]);

}

}

------------------------file read with command line args ---------------

#include <stdio.h>

main(int argc, char\*\* argv) {

if(argc < 2 )

{

printf("At least one argument required");

printf("%s filename",argv[0]);

return;

}

FILE \*fp;

char buff[1255];

fp = fopen(argv[1], "r");

fscanf(fp, "%s", buff);

printf("1 : %s\n", buff );

fclose(fp);

}

|  |  |  |
| --- | --- | --- |
| printf | Prints to screen |  |
| fprintf | To file |  |
| scanf | From keyboard |  |
| fscanf | From file |  |
| puts | Un formatted output |  |
| fputs | Un formatted file |  |
| gets | Read un formatted |  |
| fgets | Read un formatted from file |  |
|  |  |  |
|  |  |  |

------------------------------File Copy Program --------------------------

#include <stdio.h>

main(int argc, char \*argv[]) {

FILE \*fp1,\*fp2; char buff[1000]; char ch;

if(argc != 3 )

{

printf("At least two arguments required");

printf("%s sourcefile destfile \n",argv[0]);

return;

}

fp1 = fopen(argv[1], "r");

fp2 = fopen(argv[2], "w");

c = fgetc(fp1);

while (c != EOF)

{

fputc(c, fp2);

c = fgetc(fp1);

}

printf("File Copied");

fclose(fp1);// close file

fclose(fp2);// close file

}

-----------------------file copy with error check ---------------------

#include <stdio.h>

main(int argc, char \*argv[]) {

FILE \*fp1,\*fp2; char buff[1000]; char ch;

if(argc != 3 )

{

printf("At least two arguments required");

printf("%s sourcefile destfile \n",argv[0]);

return;

}

fp1 = fopen(argv[1], "r");

if(fp1 == NULL)

{

printf("Source File Could Not be opened ");

}

fp2 = fopen(argv[2], "w");

if(fp2 == NULL)

{

printf("Cannot create destination File");

}

c = fgetc(fp1); // start reading

while (c != EOF)

{

fputc(c, fp2);

c = fgetc(fp1);

}

printf("File Copied\n");

fclose(fp1);// close file

fclose(fp2);// close file

}

------------------------------binary file write -----------------

#include <stdio.h>

#include <stdlib.h>

struct threeNum

{ int n1, n2, n3;

};

int main()

{

int n;

struct threeNum num;

FILE \*fptr;

if ((fptr = fopen("bcontent","wb")) == NULL){

printf("Error! opening file");

// Program exits if the file pointer returns NULL.

exit(1);

}

for(n = 1; n < 5; ++n)

{

num.n1 = n;

num.n2 = 5\*n;

num.n3 = 5\*n + 1;

fwrite(&num, sizeof(struct threeNum), 1, fptr);

}

fclose(fptr);

return 0;

}

-------------------- 23-02-2021 -----------------------

-----------------struct binary write to file -----------

#include <stdio.h>

#include <string.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

int main () {

FILE \*fp;

fp = fopen("data", "wb");

ll n1;

n1.data = 100 ;

n1.link = 12345;

fwrite(n1, sizeof(n1), 1, fp);

printf("Done writing to file ");

fclose(fp);

return(0);

}

----------------------struct binary read from file ----------------

#include <stdio.h>

#include <string.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

int main () {

FILE \*fp;

ll n2;// empty node

fp = fopen("data", "rb");

fseek(fp, 0, SEEK\_SET);

fread(&n2, sizeof(n2), 1, fp);// fill n2 from file

printf("%d",n2.data);

printf("%u",n2.link);

fclose(fp);

return(0);

}

-----------------------------write multiple records to file

#include <stdio.h>

#include <string.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

int main () {

FILE \*fp;

fp = fopen("data", "wb");

ll n1;

n1.data = 100 ;

n1.link = 2222;

fwrite(&n1, sizeof(n1), 1, fp);

n1.data = 200 ;

n1.link = 3333;

fwrite(&n1, sizeof(n1), 1, fp);

n1.data = 200 ;

n1.link = 4444;

fwrite(&n1, sizeof(n1), 1, fp);

printf("Done writing to file ");

fclose(fp);

return(0);

}

-------------------------------read second record from file using fseek -------------

#include <stdio.h>

#include <string.h>

struct node

{

int data;

struct node \*link;

};

typedef struct node ll;

int main () {

FILE \*fp;

ll n2;// empty node

fp = fopen("data", "rb");//open file in read and binary

fseek(fp, (sizeof(n2)), SEEK\_SET);// will set file pointer to start of second record

fread(&n2, sizeof(n2), 1, fp);// fill n2 from file

printf("%d",n2.data);

printf("%u",n2.link);

fclose(fp);

return(0);

}

-----------------------Fork Demo

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int pid=0;

pid = fork();

if(pid == 0){

printf("In Child");}

else

printf("In Parent process id of child is %d ");

return 0;

}

-------------------------------- 24-02-2021 --------------------

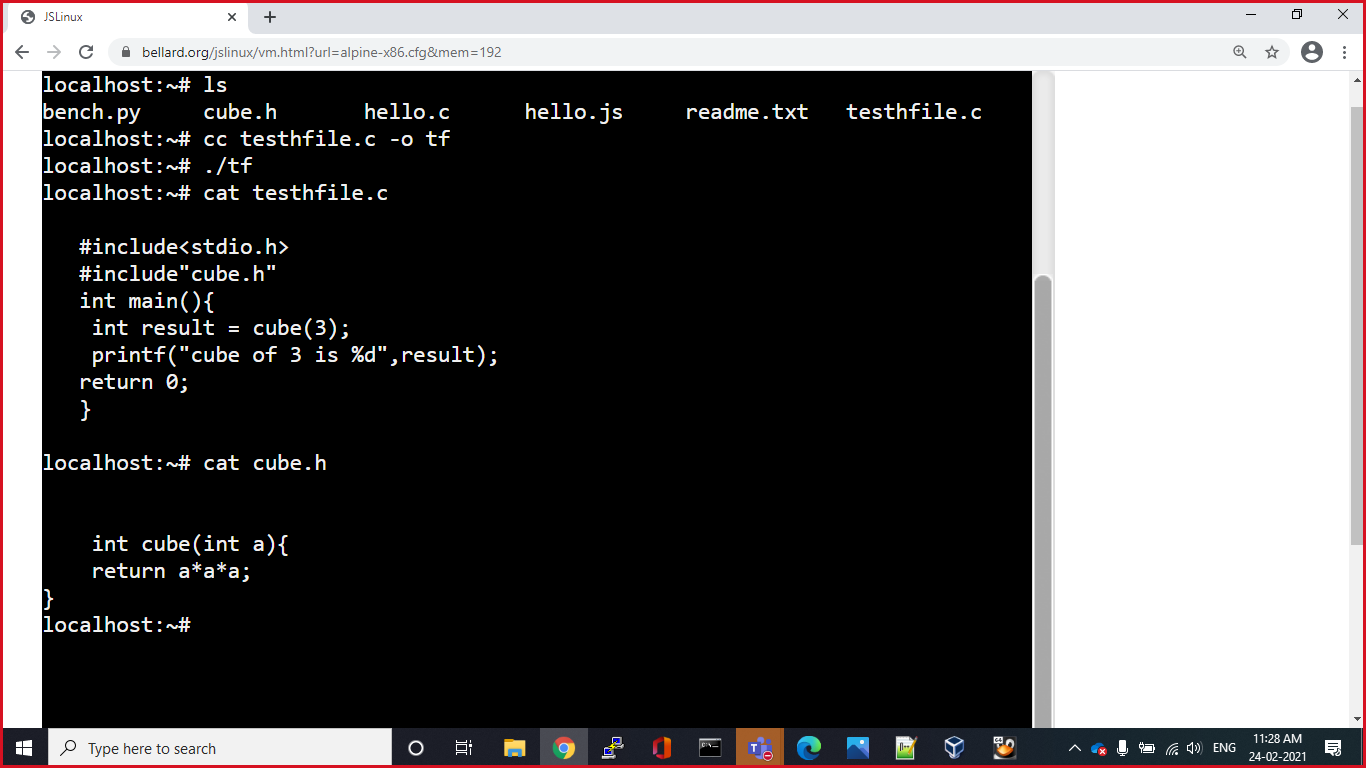
Hashing --

Log --

Header file

System call vs library call

Ping , trace , ifconfig ,



----------------------------

<https://github.com/krishnacds/linux.git>

----------------two way communication from parent to child to parent -------------------

#include<stdio.h>

#include<unistd.h>

// 2 way -- parent to Child to Parent

int main() {

int pipefds1[2]; // pipefds1[0]-->read ,,,,, pipefds1[1]-->write

int pipefds2[2]; // pipefds2[0]-->read ,,,,, pipefds2[1]-->write

char data[5]=" ";

char data1[5]=" ";

int pid=0;

if(pipe(pipefds1) < 0){printf("Pipe Error "); return;}

if(pipe(pipefds2) < 0){printf("Pipe Error "); return;}

if( (pid = fork() ) < 0) {printf("Fork Error "); return; }

//code for chlild

if(pid == 0){

sleep(5);

read(pipefds1[0],data,5); // pipefds1[0]

printf("\n %d data is %s \n",getpid(),data);

write(pipefds2[1],"HELLO",5); // pipefds2[1]

}

if(pid > 0){ // parent

write(pipefds1[1],"hello",5); // pipefds1[1]

sleep(10);

read(pipefds2[0],data1,5); // pipefds2[0]

printf(" \n %d data is %s \n",getpid(),data1);

}

//code for parent

return 0;

}

------------------------------------------ 25-02-2021 ------------------------------------

Q1)Write a C program to send "Hello" From Parent Process to Child Process

Q2)Write a C program to send "hello" form Parent to Process to Child process

and child should send "HELLO" back to Parent process

Q3)Write a C program to send 4 and 5 form Parent to child process

and child should send back sum to Parent process

Q4)Write a C program to create array of linked list. User input

1)size of array

2)size of the each linked list

------------------------25-02-2021 ----------------

Q5)Create 2 d array with user input

1)User enters no of rows

2)User enters no of cols in each row

Q6)