Experiment 1 Date:21.09.2023

Advanced use of gcc

Aim:

Advanced use of gcc : Important Options -o, -c, -D, -l, -I, -g, -O, -save-temps, -pg . Write a C program to add two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate output using gcc command and its important options.

Program

```
//sum of two numbers
#include<stdio.h>
void main(){
   int sum,a,b;
   printf("Enter first number : ");
   printf("Enter second number : ");
   scanf("%d",&a);
   scanf("%d",&b);
   sum=a+b;
   printf("sum is= %d ",sum);
}
```

GCC

GCC is a Linux- Foundationwhich is usually operated via the command line. It often comes distributed freelywith a Linux installation, so if you are running UNIX or a Linux variant you will probably have it on your system You can invoke GCC on a source code file simply by typing:-

gcc filename

The default executable output of GCC is "a.out", which can be run by typing“./a.out”. It is also possible to specify a name for the executable file at the command line by using the syntax “ -o outputfile", as shown in the following example: -

gcc filename -o outputfile

Again, you can run your program with "./outputfile". (The ./ is there to ensure to run the program for the current working directory.)

Note: If you need to use functions from the math library (generally functions from math.h” such as sin or sqrt), then you need to explicitly ask it to link with library with the “ –l ” flag and the library “m”:

gcc filename -o outputfile -lm

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sum.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
Enter first numbers : 7
```

Enter second numbers : 5
sum is= 12

Important Options in GCC

Option: -o

To write and build output to output file.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sum.c -o sum_out
Here, GCC compiles the sum.c file and generates an executable named sum_out.

Option: -c

To compile source files to object files without linking.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -c sum.c
This will generate an object file sum.o that can be linked separately.

Option: -D

To define a preprocessor macro.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -D debug=1 sum.c
This defines the macro ‘DEBUG’ with the value 1, which can be used in the source code.

Option: -l

To include a directory of header files.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -o sum.c sum_out.c -lm
Here, the -lm option links the math library (libm) with the sum.c.

Option: -I

To look in a directory for library files.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -o sum.c sum_out.c -I./ads_lab
This tells GCC to look for header files in the ads_lab directory.

Option: -g

To debug the program using GDB.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -g sum.c -o sum_out
This compiles sum.c with debug information, enabling you to debug the resulting executable.

Option: -O

To optimize for code size and execution time.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -O3 -o my_pgm sum.c
This compiles sum.c with a high level of optimization.

Option: -pg

To enable code profiling.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -pg -o my_pgm source.c
This compiles source.c with profiling support, allowing you to use profilers like gprof.

Option: -save-temps

To save temporary files generated during program execution.

Output

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -save-temps -o my_pgm source.c
This will generate intermediate files, like sum.i (pre-processed source) and sum.s (assembly code), in addition to the final executable.

Experiment 2 Date:21.09.2023

Familiarisation with GDB

Aim

Familiarisation with gdb: Important Commands - break, run, next, print, display, help. Write a C program 'mul.c' to multiply two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate sum.out which is then debug with gdb and commands.

Program

```
//product of two numbers
#include<stdio.h>
void main(){
int multi,a,b;
printf("Enter first number : ");
printf("Enter second number : ");
scanf("%d",&a);
scanf("%d",&b);
multi=a*b;
printf("product= %d ",multi);
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -g mul.c -o mul_out
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gdb mul_out
GNU gdb (Ubuntu 12.0.90-0ubuntu1) 12.0.90 Copyright (C) 2022 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details. This GDB was configured as "x86_64-linux-gnu". Type "show configuration" for configuration details. For bug reporting instructions, please see: . Find the GDB manual and other documentation resources online at: <http://www.gnu.org/software/gbd/documentation/>
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum1...
(gdb) run

Starting program: /home/mits/Desktop/s1mca/sum1 [Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Enter first numbers : 10
Enter second numbers : 3
```

Product : 30 [Inferior 1 (process 23588) exited normally]
(gdb) quit

Important Commands in GDB

Command: break

Sets a breakpoint on a particular line.

Output

(gdb) break mul.c:5

Command: run

Executes the program from start to end.

Output

(gdb) run

Command: next

Executes the next line of code without diving into functions.

Output

(gdb) next

Command: print

Displays the value of a variable.

Output

(gdb) print a

(gdb) a 10

Command: display

Displays the current values of the specified variable after every step.

Output

(gdb) display a 10

Experiment 3 Date:29.09.2023

Familiarisation with gprof

Aim

Write a program for finding the sum of two numbers using function. Then profile the executable with gprof.

Program

```
//sum of two numbers
#include<stdio.h>
int sum(int a, int b)
{
  int sum=a+b;
  return sum;
}
void main(){
    int sum,a,b;
    printf("Enter first number : ");
    printf("Enter second number : ");
    scanf("%d",&a);
    scanf("%d",&b);
    printf("sum is: %d ",sum(a,b));
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sum.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out sum.c
Enter first number:4
Enter first number:6
Sum : 10

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc -o sum.out -pg sum.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$./sum.out
Enter first number:4
Enter first number:6
Sum : 10

mits@mits-Lenovo-S510:~/Desktop/s1mca$ gprof ./sum.out gmon.out >
pgm3.txt
Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated
```

% time	cumulative seconds	self seconds	calls	self Ts/call	total Ts/call	name
0.00	0.00	0.00	1	0.00	0.00	sum

Experiment 4 Date: 29.09.2023

Different types of functions

Aim

4. Write a program for finding the sum of two numbers using different types of functions.

Algorithm:

main()

1. Start
2. Declare n1,n2,ch
3. Display choices.
4. Read option ch.
    a. if ch==1 call sum1().
    b. if ch==2 input n1 and n2 and call sum2().
    c.if ch==3 print sum3().
    d. if ch==3 input n1 and n2 and print sum4().
5. Repeat steps 3 while ch>0&&ch
6. Stop.

void sum1()

1. Start
2. Declare a and b.
3. Read a and b.
4. Print a+b.
5. Exit.

void sum2(int n1, int n2)

1. Start
2. Print a+b.
3. Exit.

int sum3()

1. Start
2. Declare a and b.
3. Read a and b.
4. Return a+b.
5. Exit.

int sum4()

1. Start
2. Return a+b

3. Exit.

Program

```
#include<stdio.h>

void sum1()
{ int a,b;
printf("Enter first & second numbers : ");
scanf("%d %d",&a,&b);
printf("Sum is: %d",a+b);
 }

void sum2(int a, int b)
{ printf("Sum is: %d",a+b);
}

int sum3()
{ int a,b;
printf("Enter first & second numbers : ");
scanf("%d %d",&a,&b);
return a+b;
}

int sum4(int a, int b)
{ return a+b;
}

void main()
{ int ch,a,b;
 do
{
printf("1. Function without return type and arguments\n2. Function without return type and
with arguments\n3. Function with return type and without arguments\n4. Function with return
type and arguments\n5. Exit\nEnter your choice: ");
scanf("%d", &ch);
switch(ch){
case 1: sum1();
        break;
case 2: printf("Enter first & second numbers : ");
        scanf("%d %d",&a,&b);
        sum2(a,b);
        break;


case 3: printf("Sum is: %d",sum3());
        break;
```

```
case 4: printf("Enter first & second numbers : ");
        scanf("%d %d",&a,&b);
        printf("Sum is: %d",sum4(a,b));
        break;
        }
 }while(ch>0&&ch<4);
}
```

Output

```
 mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc PGM1.c
 mits@mits-Lenovo-S510:~/Desktop/s1mca$./a.out PGM1.c

1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 1
Enter first & second numbers : 9 6
Sum : 15


1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 2
Enter first & second numbers : 1 8
Sum is: 9



1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 3
Enter first & second numbers : 23 25
Sum is: 48


1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
```

```
4. Function with return type and arguments
5. Exit
Enter your choice: 4
Enter first & second numbers : 40 60
Sum is: 100


1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 5
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 5 Date: 06.10.2023

Array Operations

Aim

5. To implement a menu driven program to perform following array operations

i. Insert an element to a particular location

ii. Delete an element from a particular location

iii. Traverse

Algorithm:

1.start

2.declare a[10],i,j,k,n,size=10,item,ch

3.read n

4.for i=0 to i<n{

read a[i]

increment i by 1}

5.for i=0 to i<n{

print a[i]

increment i by 1}

6.repeat until ch<5

7.read ch

8.switch the value of ch and match with cases

9.for case 1

10.read k(position to insert element)

11.read item

12.if size==n

Print “not possible to insert ’overflow’ “

13.set j=n-1

14.repeat until j>=k

{

set a[j+1]=a[j]

set j=j-1

}

15.a[k]=item

16.set n=n+1

17.for i=1 to i<n

print a[i]

18.for case 2

19.read k

20.set item=a[k]

21.repeat for j=k to n-1

{

Set a[j]=a[j+1]

}

22.set n=n-1

23.print “elements of array after deletion”

24.for i=0 to i<n

print a[i]

25.for case 4

26.exit

27.if any case does not work the default statement is printed

28.stop

Program

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
int a[10],i,j;
int n,size=10,k;
int item;
int ch;
printf("Enter the number of elements in array\n");
scanf("%d",&n);
```

```
printf("Enter elements:\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf(" \n Array is:\n");
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
do
{
printf(“MENUDRIVEN”);
printf("1.insert\n");
printf("2.delete\n");
printf("3.traverse\n");
printf("4.exit\n");
printf("enter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("Enter the position to insert:\t");
scanf("%d",&k);
printf("\nEnter the element to insert :\t");
scanf("%d", &item);
if(size==n)
{
printf("not possible to insert 'overflow' ");
}
j=n-1;
do
{
a[j+1]=a[j];
j=j-1;
}while(j>=k);
a[k]=item;
n=n+1;
printf(" After insertion array elements are:");
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
break;
case 2:
printf("Enter the position of the element to delete:\t");
```

```
scanf("%d",&k);
item=a[k];
for(j=k;j<n-1;j++)
{
a[j]=a[j+1];
}
n=n-1;
printf("After deletion array elements are:\n");
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
break;
case 3:
printf("Array elements are: \n");
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
break;
case 4:
exit(0);
}
}while(ch<5);

}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc arrayop.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out

Enter the number of of elements: 5
Enter the elements:

5

7

3

9

1
Array is:

5 7 3 9 1
MENU DRIVEN
1.insert
```

```
2.delete
3.traverse
4.exit
Enter your choice
1
Enter the position to insert : 4
Enter the item to be inserted : 6
After insertion array elements are: 5 7 3 9 6 1
MENU DRIVEN

1.insert
2.delete
3.traverse
4.exit
Enter your choice
2
Enter the position of the element to delete: 4
After deletion array elements are: 5 7 3 9 1

MENU DRIVEN1.insert
2.delete
3.traverse
4.exit
Enter your choice
3
Array elements are: 5 7 3 9 1
MENU DRIVEN

1.insert
2.delete
3.traverse
4.exit
Enter your choice
4
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 6 Date: 06.10.2023

Sort

Aim

Program to sort an integer array

Algorithm:

void main()

1.start

2.declare arr[10],i,n

3.read n

4.for i=1 to i<n

5.read arr[i]

6.increment i by 1

7. for i=1 to i<n

8.print a[i]

9. increment i by 1

10.call bubblesort(arr,n)

11.print sorted array

12.stop

void bubblesort(int arr[],int n)

1.declare i,j temp

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

if (arr[j] > arr[j + 1])
temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;

[End If]

[End for]

[End for]

3.exit

Program

```
#include <stdio.h>
void bubblesort(int arr[], int n) {
int i, j,temp;
for (i = 0; i < n - 1; i++) {
for (j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;  }
}
}
}
void main()
{
int arr[10], n, i;
printf("Enter the number of elements in the array:\n");
scanf("%d",&n);

for(i=0;i<n;i++)
{
scanf("%d",&arr[i]);
}
printf(" \n Array elements are : \n");
for(i=0;i<n;i++)
{
printf("%d\t",arr[i]);
}
bubblesort(arr, n);
printf("\n Sorted array is: ");
for ( i = 0; i < n; i++) {
printf("%d  ", arr[i]);
}

}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sort.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out
Enter the number of elements in the array: 6
7

3

9

1

6

12

Array elements are:
7 3 9 1 6 12
Sorted array is: 1 3 6 7 9 12

mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 7 Date: 06.10.2023

Search

Aim

Program to implement linear search and binary search

Algorithm:

Main()

1. start
2. declare a[100],n,j,temp,i
3. Read no of elements
4. Read array elements
5. linear(a,n);
6. binary(a,n);
7. stop

int linear(int *a, int n)

1. Declare i,f,item
2. Print 'LINEAR SEARCH'
3. Read the item to search
4. Repeat step 5 while i<n
5. if (item==la[i])
   Set flag=1
   Go to 6
   else
   Set i=i+1
6. if (f==1)
   Print "Search is successful and element is found at index", i
   else
   Print "Not found''
7. Exit

void binary (int *a,int n)

1. Declare mid,item,pos=-1,lb=0,ub=n;
2. Print 'BINARY SEARCH'
3. Read item you want to search
4. Repeat steps 5 and 6 while lb<=ub
5. set mid = (~~i+ n~~lb+ ub)/2
6. if (a[mid] == item)
   {
   pos=mid;

```
        }
        else if (a[mid] >item)
        {
        ub= mid – 1
        }
        else
        {
        lb = mid + 1
        }
        [end of if]
        [end of loop]
```

7. if(pos==-1)
   {
   print “item not present in the array"
   }
   else
   {
   Print “ item present at position, pos”
   }
   [end of if]
8. Exit

Program

```
#include<stdio.h>
#include<stdlib.h>
void binary(int *a,int n);
int linear(int *a,int n);
int main()
{
int a[100],n,j,temp,i;
printf("Enter number of elements in the array:");
scanf("%d",&n);
   for(i=0;i<n;i++)
   {
   scanf("%d",&a[i]);
   }
   printf("Array elements are:");
   for(i=0;i<n;i++)
   {
   printf("%d ",a[i]);
   }

   linear(a,n);
   binary(a,n);
   return 0;
  }
```

```
int linear(int *a,int n)
   {
   int i,f,item;
   printf("LINEAR SEARCH");
   printf("Enter the item you want to search:");
   scanf("%d",&item);
   i=0;
   f=0;
   while(i<n)
   {
   if(item==a[i])
   {
   f=1;
   break;
   }
   else
   {
   i=i+1;
   }
   }
   if(f==1)
   {
    printf("\nSearch is successful and element is found at index:%d \n",i);
   }
   else
   {
   printf("Elemnent not found");
   }
   return 0;
   }
 void binary(int *a,int n)
 {
 int mid,item,pos=-1,lb=0,ub=n;
 printf("BINARY SEARCH");
 printf("\nEnter the item you want to search:");
 scanf("%d",&item);
 while(lb<=ub)
    {
      mid=(lb+ub)/2;
      if(a[mid]==item)
         {
           pos=mid;

           break;
         }
      else if(a[mid]>item)
         {
           ub=mid-1;
         }
      else
```

```
{
        lb=mid+1;
       }
   }
 if(pos==-1)
 {
 printf("Item not present in the array");
 }
 else
 {
 printf("Item present at position:%d",pos);
 }

 }
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc serch.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out
Enter number of elements in the array:5
13

15

19

12

11
Array elements are:13 15 19 12 11

LINEAR SEARCH

Enter the item you want to search:19
Search is successful and element is found at index:2
BINARY SEARCH
Enter the item you want to search:15
Item present at position:1

 mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 8 Date: 06.10.2023

Matrix

Aim

Perform addition, subtraction and multiplication of two matrices

Algorithm:

1. declare int a[2][2],b[2][2],i,j,c;
2. declare int sum[2][2],sub[2][2],mult[2][2];
3. Read the first matrix
4. Display the first matrix
5. Read the second matrix
6. Display the second matrix
7. Display 1:addition 2:subtraction 3:multiplication 4:exit , Enter your choice
8. Repeat the steps 9 to 13 while c<4
9. Read choice c.
10. if c==1

```
for(i=0;i<2;i++){
for(j=0;j<2;j++){
sum[i][j]=a[i][j]+b[i][j];
}
}
for(i=0;i<2;i++){
for(j=0;j<2;j++){
display matrix after addition
```

11. if c==2

```
for(i=0;i<2;i++){
for(j=0;j<2;j++){
sub[i][j]=a[i][j]-b[i][j];
}
}
for(i=0;i<2;i++){
for(j=0;j<2;j++){
display matrix after subtraction
```

12. if c==3

```
for(i=0;i<2;i++){
for(j=0;j<2;j++){
mult[i][j]=a[i][j]*b[i][j];
}
}
for(i=0;i<2;i++){
for(j=0;j<2;j++){
display matrix after multiplication
```

13. if c==4

display exit

Program

```
#include<stdio.h>
int main(){
int a[2][2],b[2][2],i,j,c;
int sum[2][2],sub[2][2],mult[2][2];
printf("Enter the first matrix:");
for(i=0;i<2;i++)
for(j=0;j<2;j++)
scanf("%d",&a[i][j]);

for(i=0;i<2;i++){

for(j=0;j<2;j++)
{
printf("%d\t",a[i][j]);}
printf("\n");
}
printf("Enter the second matrix:");
for(i=0;i<2;i++)
for(j=0;j<2;j++)
scanf("%d",&b[i][j]);
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("%d\t",b[i][j]);}
printf("\n");
}
Do
{
printf("\n1: Addition\n2: Subtraction\n3: Multiplication\n4=Exit\n Enter your choice:");
scanf("%d",&c);
switch(c)
{
case 1:
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
sum[i][j]=a[i][j]+b[i][j];
}
}
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("%d\t",Sum[i][j]);
}
printf("\n");
```

```
}
break;
case 2:
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
sub[i][j]=a[i][j]-b[i][j];
}
}
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("%d\t",Sub[i][j]);
}
printf("\n");
}
break;
case 3:
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
mult[i][j]=a[i][j]*b[i][j];
}
}
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
printf("%d\t",Mult[i][j]);
}
printf("\n");
}
break;
default:
printf("Exit");
break;
}
}while(c<4);
return 0;
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc matrix.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out
Enter the first matrix:5 4 3 2
5       4
3       2
Enter the second matrix:1 8 9 10
1       8
9       10

1: Addition
2: Subtraction
3: Multiplication
4: Exit
Enter your choice:1
6       12
12      12

1: addition
2: subtraction
3: multiplication
4: exit
Enter your choice:2
-4      -4
-6      -8

1: addition
2: subtraction
3: multiplication
4: exit
Enter your choice:3
41      80
21      44

1: addition
2: subtraction
3: multiplication
4: exit
Enter your choice:4
Exit
```

Experiment 9 Date: 12.10.2023

Stack Operations

Aim

Program to implement stack operations using arrays.

Algorithm:

main()

1. start
2. declare and initialise A[10],i,ch,item,size,top=-1
3. repeat the steps 4 to 7 until while(ch!=3)
4. display 1. For push 2. For pop 3. for exit and Enter your choice
5. if(ch==1)
   {
   Read the item
   Top=push(A,top,item,size)
   Display elements of the stack
   }
6. if(ch==2)
   {
   Top=pop(A,top,item);
   Display elements of the stack
   }
7. If (ch==3)
   {
   Exit
   }
8. Stop

int push(int *A,int top,int item,int size)

1. declare i
2. If(top=size-1)
   print 'Stack is full'
   else
   top=top+1
   A[top]=item
   print top
   display inserted item
   return top
3. Exit

int pop(int*A,int top,int item)

1. if(top<0)

```
        print 'Stack is empty'
        else
            item=A[top]
            top=top-1
```

2. Print popped item
3. Return top
4. Exit

Program

```
#include<stdio.h>
#include<stdlib.h>
int push(int*A, int top,int item,int size)
{
int i;
if(top==size-1)
{
printf("Stack is full");
}
else
{
 top=top+1;
 A[top]=item;
 printf("%d",top);
  }
  printf("Inserted item is%d:",item);
  return top;
  }

int pop(int*A,int top,int item)
{
if(top<0)
{
 printf("Stack is empty");
 }
 else
 {
 item=A[top];
 top=top-1;
 }
  printf("Popped item is  %d", item) ;
  return top;
  }
  void main()
  {
  int ch,top=-1,item,A[10],size,i ;
```

```
do
{
printf("\n1 for push\n2 for pop\n3 for exit\n Enter your choice :");


scanf("%d",&ch);
switch(ch)
{
case 1:
printf("Enter the item : ");
scanf("%d",&item);
top=push(A,top,item,size);
printf("Stack elements are: ");
for(i=0;i<top+1;i++)
{
printf("%d\t",A[i]);
}
break;
case 2:
top=pop(A,top,item);
printf("Stack elements are:");
for(i=0;i<top+1;i++)
{
printf("%d\t",A[i]);
}
break;
case 3:
exit(0);
default:
printf("Exit");
}
}
While(ch!=3)
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc stack.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out
 for push
 for pop
 for exit
Enter your choice :1
Enter the item : 3
Inserted item is 3
Stack elements are: 3
```

```
 for push
 for pop
 for exit
Enter your choice :1
Enter the item : 8

Inserted item is 8
Stack elements are: 3 8
for push
for pop
for exit
Enter your choice :2
Popped item is  8
Stack elements are:22
for push
for pop
for exit
Enter your choice :3
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 10 Date: 12.10.2023

Queue Operations

Aim

Program to implement queue operations using arrays

Algorithm:

Int main()

1: declare front,rear,size,q[5]
2: declare item,choice,i
3: display 1.Insertion 2.deletion 3.traversal 4.exit
4: reapeat the step 5 to 8 while(choice>0 && choice<3)
5: read choice
6: if(choice==1)
read item
call enqueue(item)
7: if(choice==2)
item=call dequeue()
8: if(choice==3)
Display queue elements
i=front
while(i<rear)
dispalay q[i]
i++
9: if(choice==4)
goto step 10
10: stop

void enqueue(item**)**

1: if rear= size- 1
Write overflow
Go to step4
2: else
if front = -1 and rear = -1
front = rear = 0
else
rear = rear + 1
3: q[rear] = value
4: Exit

int dequeue()

1: if front = -1 and rear=-1
Display Queue is empty

```
    goto step3

2:  else
      item = q[front]
      if  front=rear
      front=rear=-1
      else
      front = front + 1
3:return item
```

Program

```
#include<stdio.h>
int front=-1,rear=-1,size=5,q[5];
void enqueue(int item){
        if(rear== size-1){
           printf("queue overflow"); }
        else{
          if(front==-1 && rear==-1)
             front=rear=0;
          else{
             rear=rear+1;
            q[rear]=item;
        }
         }
  int dequeue(){
        int value;
        if(front==-1 && rear==-1)
           printf("queue underflow");
        value=q[front];
        if(front==rear)
          front=rear=-1;
        else
           front=front+1;
        return value; }
  int main(){
        int item,i,choice;
        printf("\n1.insertion\n2.deletion\n3.traversal\n4.exit\n");
        do{
              printf("\nEnter the choice");
          scanf("%d",&choice);
          if(choice==1){
                  printf("Enter the element to insert");
                  scanf("%d",&item);
```

```
                enqueue(item);}

        if(choice==2){
                item=dequeue();
                printf("Deleted item:%d",item);}
          if(choice==3){
                printf("Queue elements are:");
                for(i=front;i<=rear;i++)
                 printf("%d\t",q[i]);}
          if(choice==4){
                printf("Exit");  }
         }while(choice>0 &&choice<3);
       return 0;
       }
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc queue.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out

 1.insertion
 2.deletion
 3.traversal
 4.exit
Enter your choice 1
Enter the element to insert 7
Enter your choice 1
Enter the element to insert 9
Enter your choice 1
Enter the element to insert 11
Enter your choice 3
Queue elements are:
7 9 11
Enter your choice 2
Deleted item: 7
Enter your choice 3
Queue elements are:
9 11
Enter your choice 4
Exit
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```