**EXPERIMENT – 11**

**AIM :**  Create an array having information about a student and perform

a. Insert a new student at specified position.

b. Delete of a student with the roll number of student specified.

c. Reversal of that array.

**ALGORITHM :**

**SOURCE CODE:**

#include <bits/stdc++.h>

using namespace std;

class Student {

public :

    string name;

    int rollNo;

    Student(string name,int rollNo) {

        this->name=name;

        this->rollNo=rollNo;

    }

    Student() {

    }

};

void disp(Student arr[],int size) {

    cout << "[ ";

    for (int idx=0;idx<size;idx++) {

        cout << "{ " << arr[idx].name << "," << arr[idx].rollNo << " }";

        if (idx!=size-1) {

            cout << ",";

        }

    }

    cout << " ]" << endl;

}

void insertion(Student arr[],string name,int rollNo,int pos,int size) {

    for (int idx=size;idx>pos;idx--) {

        arr[idx]=arr[idx-1];

    }

    arr[pos]=Student(name,rollNo);

    disp(arr,size+1);

}

int deletion(Student arr[],int pos,int size) {

    int deleted=arr[pos].rollNo;

    for (int idx=pos;idx<size-1;idx++) {

        arr[idx]=arr[idx+1];

    }

    disp(arr,size-1);

    return deleted;

}

void reverse(Student arr[],int size) {

    int s=0;

    int e=size-1;

    while (s<e) {

        Student store=arr[s];

        arr[s]=arr[e];

        arr[e]=store;

        s++;

        e--;

    }

    disp(arr,size);

}

int main() {

    int sizeLimit;

    cout << "Enter the size limit of students : " << endl;

    cin >> sizeLimit;

    Student arr[sizeLimit];

    bool want=true;

    string name;

    int rollNo,idx,size=0;

    while (want) {

        cout << "Do you want to insert a new student in array ?" << endl;

        cin >> want;

        if (want) {

            cout << "Enter the name,roll no. and index of student" << endl;

            cin >> name >> rollNo >> idx;

            if (idx>size) {

                cout << "Invalid index !!" << endl;

                continue;

            }

            if (size==sizeLimit) {

                cout << "Size limit exceeded !!" << endl;

                continue;

            }

            insertion(arr,name,rollNo,idx,size);

            size++;

            continue;

        }

        cout << "Do you want to delete a student from array ?" << endl;

        cin >> want;

        if (want) {

            cout << "Enter the index of student" << endl;

            cin >> idx;

            if (idx>=size) {

                cout << "Invalid index !!" << endl;

                continue;

            }

            rollNo=deletion(arr,idx,size);

            size--;

            cout << "Deleted student's roll no. is : " << rollNo << endl;

            continue;

        }

        cout << "Do you want to reverse the array of students ?" << endl;

        cin >> want;

        if (want) {

            reverse(arr,size);

            continue;

        }

    }

    return 0;

}

**EXPERIMENT – 11**

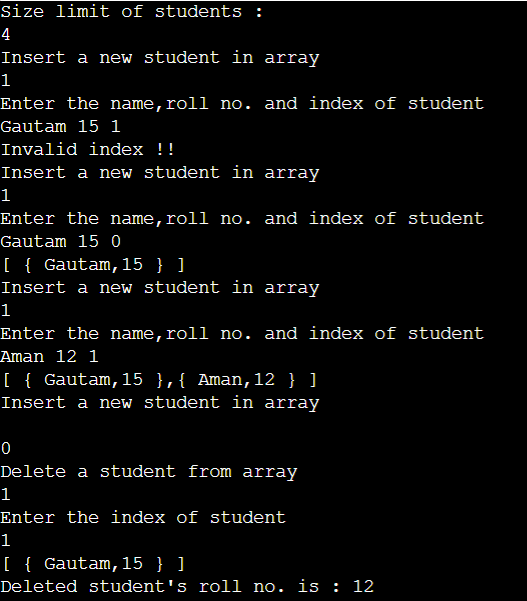
**AIM :**  Create an array having information about a student and perform

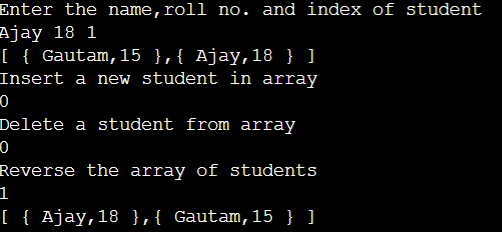
a. Insert a new student at specified position.

b. Delete of a student with the roll number of student specified.

c. Reversal of that array.

**OUTPUT :**





**EXPERIMENT – 12**

**AIM :** Create a Circular Queue using Array and implement different operations such as insert, delete and display the queue elements.

**ALGORITHM :**

**SOURCE CODE:**

#include <bits/stdc++.h>

using namespace std;

class CircularQueue {

int front=0,size=0,sizeLimit;

int\* arr;

public :

CircularQueue(int sizeLimit) {

this->sizeLimit=sizeLimit;

arr=new int[sizeLimit];

}

void enqueue(int element) {

if (size==sizeLimit) {

cout << "Queue is full !!" << endl;

return;

}

arr[(front+size)%sizeLimit]=element;

size++;

disp();

}

int dequeue() {

if (size==0) {

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

return -1;

}

int dequeued=arr[front];

front=(front+1)%sizeLimit;

size--;

disp();

return dequeued;

}

int peek() {

if (size==0) {

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

return -1;

}

disp();

return arr[front];

}

void disp() {

cout << "Queue : ";

for (int idx=front;idx<front+size;idx++) {

cout << arr[idx%sizeLimit] << " ";

}

cout << endl;

}

};

int main() {

int n;

cout << "Size limit of circular queue : ";

cin >> n;

CircularQueue queue(n);

bool want=true;

int element,dequeued,front;

while (want) {

cout << "Enqueue an element in queue " << endl;

cin >> want;

if (want) {

cout << "Enter the element to be enqueued : " << endl;

cin >> element;

queue.enqueue(element);

continue;

}

cout << "Dequeue an element from queue " << endl;

cin >> want;

if (want) {

dequeued=queue.dequeue();

if (dequeued!=-1) {

cout << "Element dequeued is : " << dequeued << endl;

}

continue;

}

cout << "Peek an element of queue " << endl;

cin >> want;

if (want) {

front=queue.peek();

if (front!=-1) {

cout << "Peek element is : " << front << endl;

}

continue;

}

}

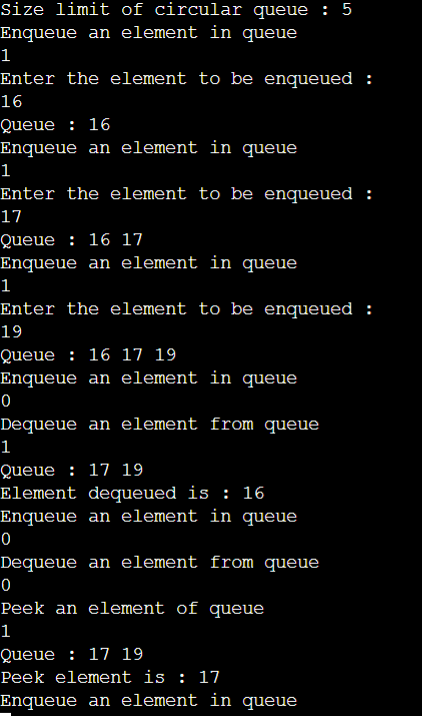
return 0;

}

**EXPERIMENT – 12**

**AIM :**  Create a Circular Queue using Array and implement different operations such as insert, delete and display the queue elements.

**OUTPUT :**

****

**EXPERIMENT-13**

**AIM :** Write a program to convert infix to postfix expression using stack.

**ALGORITHM :**

**SOURCE CODE :**

#include <bits/stdc++.h>

using namespace std;

string postfix(string infix) {

string postfix;

stack<char> s;

unordered\_map<char,int> priority={ {'+',1},{'-',1},{'/',2},{'\*',2},{'%',2},{'^',3} };

for (int idx=0;idx<infix.length();idx++) {

if (infix.at(idx)==')') {

while (s.top()!='(') {

postfix.push\_back(s.top());

s.pop(); }

s.pop();

continue; }

if (priority.count(infix.at(idx))) {

while (!s.empty() && s.top()!='(' && priority.at(s.top())>=priority.at(infix.at(idx))) {

postfix.push\_back(s.top());

s.pop(); }

s.push(infix.at(idx));

continue; }

if (infix.at(idx)=='(') {

s.push(infix.at(idx)); }

else {

postfix.push\_back(infix.at(idx)); }

while (!s.empty()) {

postfix.push\_back(s.top());

s.pop();}

return postfix;}

int main() {

string infix;

cout << "Enter the infix expression : " << endl;

cin >> infix;

cout << "Its Postfix expression is : " << endl;

cout << postfix(infix);

return 0;}

**EXPERIMENT-13**

**AIM :** Write a program to convert infix to postfix expression using stack.

**OUTPUT:**

Text

Description automatically generated