Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Стеки и очередь через классы

Выполнил работу

Студент группы РИС-22-1Б

Лихачев Д.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Постановка задачи**

Создать классы стека и очереди, которые смогут добавлять и удалять элементы, а также дополнительные функции для работы со ними.

**Код программы**

#include <iostream>

#define SIZE 10

#define \_SIZE 1000

using namespace std;

class Queue

{

private:

int\* arr;

int capacity;

int front;

int rear;

int count;

public:

Queue(int size = \_SIZE);

~Queue();

int dequeue();

void enqueue(int x);

int peek();

int size();

bool isEmpty();

bool isFull();

int\* getArr() const { return arr; }

int getCapacity() const { return capacity; }

int getFront() const { return front; }

int getRear() const { return rear; }

int getCount() const { return count; }

};

Queue::Queue(int size)

{

arr = new int[size];

capacity = size;

front = 0;

rear = -1;

count = 0;

}

Queue::~Queue() {

delete[] arr;

}

int Queue::dequeue()

{

if (isEmpty())

{

cout << "Underflow\nProgram Terminated\n";

exit(EXIT\_FAILURE);

}

int x = arr[front];

cout << "Removing " << x << endl;

front = (front + 1) % capacity;

count--;

return x;

}

void Queue::enqueue(int item)

{

if (isFull())

{

cout << "Overflow\nProgram Terminated\n";

exit(EXIT\_FAILURE);

}

cout << "Inserting " << item << endl;

rear = (rear + 1) % capacity;

arr[rear] = item;

count++;

}

int Queue::peek()

{

if (isEmpty())

{

cout << "Underflow\nProgram Terminated\n";

exit(EXIT\_FAILURE);

}

return arr[front];

}

int Queue::size()

{

return count;

}

bool Queue::isEmpty()

{

return (size() == 0);

}

bool Queue::isFull()

{

return (size() == capacity);

}

class Stack

{

private:

int\* arr;

int top;

int capacity;

public:

Stack(int size = SIZE);

~Stack();

void push(int);

int pop();

int peek();

int size();

bool isEmpty();

bool isFull();

int\* getArr() const { return arr; }

int getTop() const { return top; }

int getCapacity() const { return capacity; }

};

Stack::Stack(int size)

{

arr = new int[size];

capacity = size;

top = -1;

}

Stack::~Stack()

{

delete[] arr;

}

void Stack::push(int x)

{

if (isFull())

{

cout << "Overflow\nProgram Terminated\n";

exit(EXIT\_FAILURE);

}

cout << "Inserting " << x << endl;

arr[++top] = x;

}

int Stack::pop()

{

if (isEmpty())

{

cout << "Underflow\nProgram Terminated\n";

exit(EXIT\_FAILURE);

}

cout << "Removing " << peek() << endl;

return arr[top--];

}

int Stack::peek()

{

if (!isEmpty())

return arr[top];

else

exit(EXIT\_FAILURE);

}

int Stack::size()

{

return top + 1;

}

bool Stack::isEmpty()

{

return top == -1;

}

bool Stack::isFull()

{

return top == capacity - 1;;

}

int main()

{

Stack pt(3);

pt.push(1);

pt.push(2);

pt.pop();

pt.pop();

pt.push(3);

cout << "The top element is " << pt.peek() << endl;

cout << "The stack size is " << pt.size() << endl;

pt.pop();

if (pt.isEmpty())

cout << "The stack is empty\n";

else

cout << "The stack is not empty\n";

cout << endl;

Queue q(5);

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

cout << "The front element is " << q.peek() << endl;

q.dequeue();

q.enqueue(4);

cout << "The queue size is " << q.size() << endl;

q.dequeue();

q.dequeue();

q.dequeue();

if (q.isEmpty())

cout << "The queue is empty\n";

else

cout << "The queue is not empty\n";

return 0;

}

**Вывод программы**



***Рисунок 1 – вывод программы***



***Рисунок 2 – UML диаграмма***

**Анализ результатов**

Программа выполняет свою задачу.