**Постановка задачи.**

Реализовать стек на основе массива на с++.

**Введение.**

Стек - это абстрактный тип данных, который представляет собой список элементов, организованных по принципу LIFO (Last In First Out). Это означает, что последний добавленный элемент будет первым удаленным.

Существует несколько методов для работы со стеком. Например, push используется для добавления элемента в стек, а pop используется для удаления элемента из стека. [Еще один метод peek используется для получения значения верхнего элемента стека без его удаления](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA).

**Решение.**

#include <iostream>

using namespace std;

class stack {

public:

int array[100];

int size = -1;

//Добавление элемента в конец стека

void push(int x) {

if (size == 100) {

cout << "Stack is full" << endl;

return;

}

array[size + 1] = x;

size++;

}

//Удаление элемента из конца стека

void pop() {

if (size == -1) {

cout << "Stack is empty" << endl;

return;

}

array[size] = 0;

size--;

}

//Получение последние элемента стека

int peak() {

if (size == -1) {

cout << "Stack is empty" << endl;

return 0;

}

int toReturn=array[size];

size--;

return toReturn;

}

//Печать стека

void printQueue() {

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

cout << array[i];

if (i != size) {

cout << "; ";

}

}

cout << endl;

}

};

int main() {

//Создание экземпляра очереди

stack exampleStack;

//Добавление в конец стека

exampleStack.push(1);

exampleStack.push(2);

exampleStack.push(3);

exampleStack.push(4);

exampleStack.push(5);

//Печать стека

exampleStack.printQueue();

//Удаление из начала стека

exampleStack.pop();

//Получение элемента из конца стека

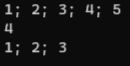
cout << exampleStack.peak() << endl;

//Печать стека

exampleStack.printQueue();

return 0;

}

****

**Заключение.**

В результате выполнения данной задачи, я успешно реализовала стек на основе массива на языке программирования C++. В результате получилось эффективное решение, которое можно использовать в приложениях, где нужно хранить и обрабатывать большие объемы данных.