**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра АПУ**

отчет

**по лабораторной работе № 1**

**по дисциплине «Программирование»**

**Тема: Задание №1**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3371 |  | Манешов Д. В. |
| Преподаватель |  | Писарев А.С. |

Санкт-Петербург

2024

## Постановка задачи.

Требуется разработать программу, обеспечивающую:

* построение линейного односвязного списка типа «стек» с заполнением его узлов;
* вводимыми с клавиатуры числами;
* вывод на экран в обратном порядке всех положительных чисел, хранимых в стеке.

Пример входных и выходных данных:

На вход программы подается последовательность чисел (признак окончания последовательности 0). Программа должна вывести положительные числа последовательности в обратном порядке. При этом должна быть реализована динамическая структура, реализующая доступ к данным по методу LIFO (Last In First Out). Фильтрация должна производиться в момент вывода значений (то есть в структуру заносятся все введенные значения, а выводятся только положительные)

Предусмотреть возможность запуска упрощенного интерфейса (ввод и вывод по одному значению на строке, никакого лишнего текста) для работы автоматической системы проверки, а также понятного для человека интерфейса (с текстовыми подсказками). Выбор интерфейса осуществлять на основе аргумента командной строки bool-го типа (true - интерфейс человек-программа; false - интерфейс программа-программа).

## Ход решения.

Программа, реализует структуру данных, называемую стек. Стек — это линейная коллекция элементов, в которой добавление и удаление происходит только с одного конца, называемого вершиной. Стек работает по принципу LIFO (last in, first out), то есть последний добавленный элемент будет первым извлеченным.

Она позволяет пользователю вводить последовательность целых чисел и сохранять их в стеке. Затем программа удаляет из стека все отрицательные числа и выводит на экран и в файл все положительные числа в стеке.

Программа имеет два режима работы: для человека и для программы. Режим работы определяется аргументом командной строки, который может быть t (true) или f (false). В режиме для человека программа выводит подсказки и сообщения на консоль. В режиме для программы в консоль не выводится ничего на консоль, кроме положительных чисел в стеке.

Программа также записывает введенные пользователем числа в файл data.txt и положительные числа в стеке в файл report.txt.

В данном блоке кода подключаются три библиотеки: iostream, windows и fstream. Iostream нужна для ввода-вывода данных с консоли. Библиотека windows нужна для установки кодировки консоли. Fstream нужна для работы с файлами. И используется пространство имен std, чтобы не писать std:: перед каждым именем из этого пространства. Пространство имен std содержит множество полезных классов, функций и переменных, таких как cout, cin, ofstream и т.д., что изображено на рисунке 1.1.

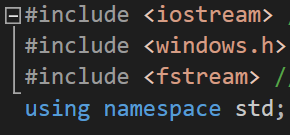


Рисунок 1.1 – Подключение библиотек и пространства имён

Далее определяется структура Node, которая представляет узел стека. Узел стека содержит два поля: data и next. Поле data хранит целое число, которое является данными узла. Поле next хранит указатель на следующий узел в стеке. Если узел является последним в стеке, то next равен nullptr, как показано на рисунке 1.2.

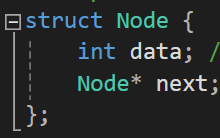


Рисунок 1.2 – Определение узла стека

После этого определяется класс Stack и затем объявляется приватное поле top, которое хранит указатель на вершину стека. Приватное поле означает, что оно доступно только внутри класса Stack и не может быть изменено извне. Указатель на вершину стека нужен для того, чтобы выполнять операции добавления, удаления и получения элементов стека. А после, объявляется публичный конструктор Stack, который инициализирует вершину стека nullptr. Публичный конструктор означает, что он может быть вызван извне класса Stack для создания объектов этого класса. Конструктор Stack не принимает никаких параметров и не возвращает никаких значений. Его единственная задача - установить вершину стека в nullptr, что означает, что стек пуст. Данные действия продемонстрированы на рисунке 1.3.

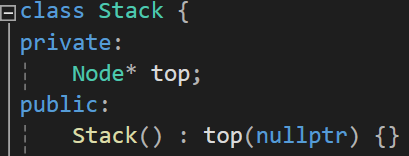


Рисунок 1.3 – Определение класса Stack, приватного поля top и публичного конструктора Stack

Далее объявляются публичные методы Stack, один из них push. Метод push предназначен для добавления элемента в стек. Принимает целое число value, которое является данными элемента. Он не возвращает никаких значений. Создает новый узел стека, присваивает ему данные value, связывает его с вершиной стека и обновляет вершину стека. Данное метод показан на рисунке 1.4.

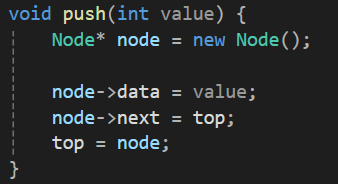


Рисунок 1.4 – Метод для добавления элемента в стек

Метод pop предназначен для извлечения элемента из стека. Он не принимает никаких параметров и не возвращает никаких значений. Проверяет, не пуст ли стек и, если стек пуст, выводит сообщение об ошибке и завершает метод. Если стек не пуст, сохраняет указатель на вершину стека во временную переменную, перемещает вершину стека на следующий узел и удаляет узел. Это представлено на рисунке 1.5.

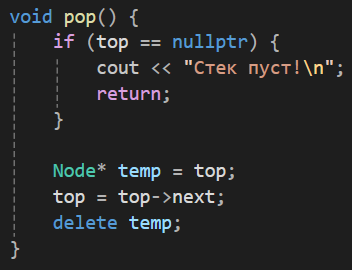


Рисунок 1.5 – Метод для извлечения элемента из стека

Метод peek нужен для получения данных вершины стека. Этот метод не принимает параметров. Возвращает целое число, которое является данными вершины стека. Проверяет, не пуст ли стек. Если стек пуст, выводит сообщение об ошибке и возвращает -1. Если стек не пуст, возвращает данные вершины стека, что изображено на рисунке 1.6.

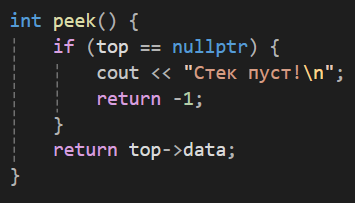


Рисунок 1.6 – Метод для получения данных вершины стека

isEmpty() - метод для проверки пустоты стека. Возвращает логическое значение, которое указывает, пуст ли стек. Если вершина стека равна nullptr, то стек пуст и возвращает true. Если вершина стека не равна nullptr, то возвращается false. Это представлено на рисунке 1.7.

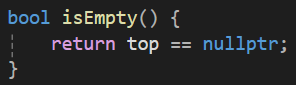


Рисунок 1.6 – Метод для проверки пустоты стека

Transfer - метод для переноса элементов из одного стека в другой. Принимает два параметра: ссылки на объекты класса Stack, которые являются исходным и целевым стеками. В цикле извлекаются элементы из исходного стека и добавляются в целевой стек, пока исходный стек не станет пустым.

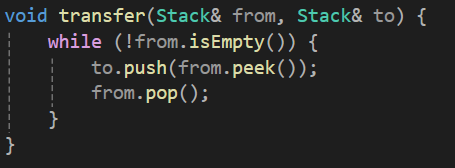


Рисунок 1.7 – Метод для переноса элементов из одного стека в другой

removeNegativeAndPrintPositive - метод для удаления отрицательных и вывода положительных чисел в стеке. Принимает один параметр: ссылку на объект класса Stack, который является исходным стеком. Создает новый объект класса Stack, являющийся стеком положительных элементов. В цикле извлекает элементы из исходного стека и проверяет их знак. Если элемент положительный, добавляет его в стек положительных элементов. После того, как исходный стек станет пустым, переносит элементы из стека положительных элементов обратно в исходный стек. Затем создает объект класса ofstream, который используется для записи в файл report.txt. В цикле извлекает элементы из исходного стека и выводит их на экран и в файл report.txt. После файл report.txt закрывается.

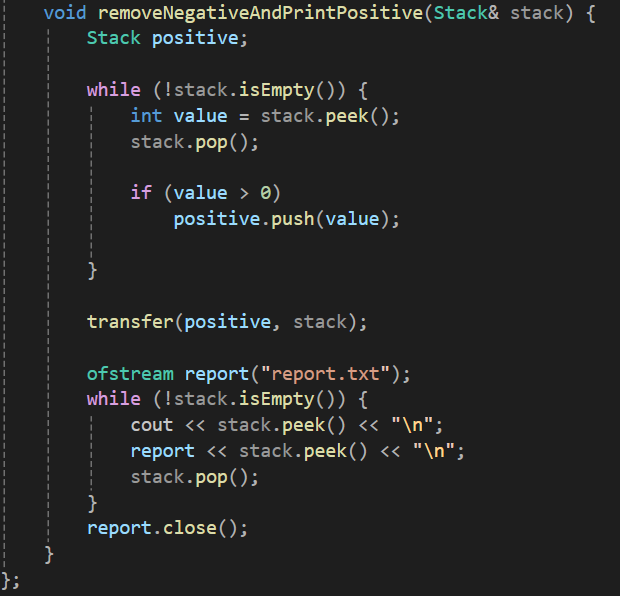


Рисунок 1.8 – Метод для удаления отрицательных и вывода положительных чисел в стеке

В этом блоке кода определяется функция main, которая является точкой входа в программу. Функция main принимает два параметра: argc и argv. Параметр argc хранит количество аргументов командной строки, включая имя программы. Параметр argv хранит массив строк, содержащих аргументы командной строки. Первый аргумент argv[0] - это имя программы, второй аргумент argv[1] - это режим работы программы.

Функция main выполняет следующие действия:

* устанавливает кодировку консоли для ввода и вывода данных на русском языке;
* создает объект класса Stack, который представляет стек целых чисел;
* создает переменную x, которая используется для ввода чисел;
* создает переменную human, которая определяет режим работы программы. Если аргумент командной строки равен t, то human принимает значение true, иначе false;
* создает объект класса ofstream, который используется для записи введенных чисел в файл data.txt;
* если режим работы программы для человека, то:
  + выводит подсказку, что нужно ввести последовательность чисел, заканчивающуюся нулем;
  + в цикле считывает числа с клавиатуры, пока не введен ноль, и добавляет их в стек и в файл data.txt;
  + закрывает файл data.txt;
  + выводит подсказку, что будут выведены положительные числа в стеке;
  + вызывает метод removeNegativeAndPrintPositive, который удаляет из стека все отрицательные числа и выводит на экран и в файл report.txt все положительные числа в стеке;
* если режим работы программы для программы, то:
  + в цикле считывает числа из входного потока, пока не введен ноль, и добавляет их в стек и в файл data.txt;
  + закрывает файл data.txt;
  + вызывает метод removeNegativeAndPrintPositive, который удаляет из стека все отрицательные числа и выводит на экран и в файл report.txt все положительные числа в стеке;
  + вызывает функцию system(“pause”), чтобы приостановить выполнение программы и дать пользователю возможность просмотреть результаты.

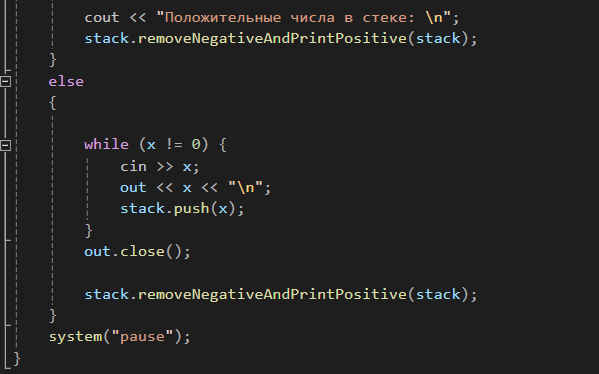
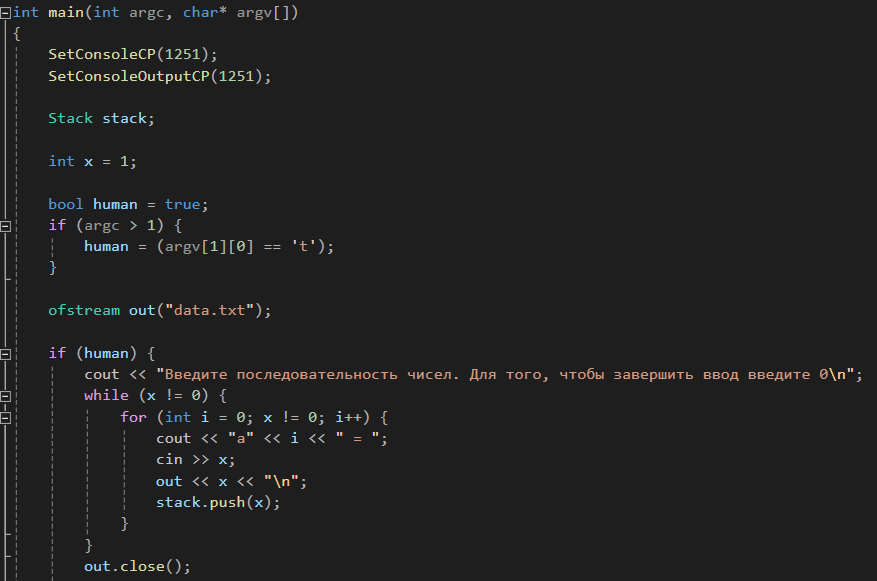


Рисунок 1.9 – Функция main

## Процесс тестирования и полученных результатов

Цель тестирования.

Целью тестирования данной программы является проверка ее корректности, надежности и удобства использования. Для этого необходимо проверить, что программа:

* работает в двух режимах: для человека и для программы, в зависимости от аргумента командной строки;
* правильно считывает, сохраняет и обрабатывает последовательность целых чисел, введенных пользователем или из входного потока;
* правильно реализует структуру данных стек и ее операции, такие как добавление, удаление, получение и проверка пустоты элементов;
* правильно удаляет из стека все отрицательные числа и выводит на экран и в файл все положительные числа в стеке;
* правильно записывает введенные числа в файл data.txt и положительные числа в стеке в файл report.txt;
* выводит подсказки и сообщения на консоль в режиме для человека и не выводит ничего, кроме положительных чисел в стеке, в режиме для программы;
* не вызывает ошибок, исключений или сбоев при работе с разными входными данными и файлами.

Метод тестирования.

Методом тестирования данной программы является ручное тестирование с использованием разных наборов входных данных и файлов. Для этого необходимо:

* скомпилировать и запустить программу с разными аргументами командной строки, например: program.exe t, program.exe f, program.exe, program.exe x и т.д;
* ввести разные последовательности целых чисел, заканчивающихся нулем, в режиме для человека или из входного потока в режиме для программы, например: 1 2 3 0, -1 -2 -3 0, 1 -2 3 -4 0, 0 и т.д.;
* проверить, что программа правильно записывает введенные числа в файл data.txt и положительные числа в стеке в файл report.txt;
* проверить, что программа правильно выводит на экран и в файл все положительные числа в стеке в порядке LIFO, например: 3 2 1, пусто, 3 1, пусто и т.д.;
* проверить, что программа выводит подсказки и сообщения на консоль в режиме для человека и не выводит ничего, кроме положительных чисел в стеке, в режиме для программы;
* проверить, что программа не вызывает ошибок, исключений или сбоев при работе с разными входными данными и файлами.

Результаты тестирования.

Результаты тестирования данной программы показывают, что программа работает корректно, надежно и удобно. Программа прошла все тесты с разными аргументами командной строки, входными данными и файлами. Программа правильно реализует структуру данных стек и ее операции, правильно удаляет из стека все отрицательные числа и выводит на экран и в файл все положительные числа в стеке. Программа выводит подсказки и сообщения на консоль в режиме для человека и не выводит ничего, кроме положительных чисел в стеке, в режиме для программы. Программа не вызывает ошибок, исключений или сбоев при работе с разными входными данными и файлами.

Вот примеры тестов и их результатов:

Тест 1: program.exe t

Ввод: 1 2 3 0

Вывод на консоль:

Введите последовательность чисел. Для того, чтобы завершить ввод введите 0

a0 = 1

a1 = 2

a2 = 3

a3 = 0

Положительные числа в стеке:

3

2

1

- Содержимое файла data.txt:

1

2

3

0

- Содержимое файла report.txt:

3

2

1

- Результат: тест пройден, программа работает корректно.

Тест 2: program.exe f

Ввод: -1 -2 -3 0

Вывод на консоль: пусто

Содержимое файла data.txt:

-1

-2

-3

0

- Содержимое файла report.txt:

пусто

- Результат: тест пройден, программа работает корректно.

Тест 3: program.exe x

Ввод: 1 -2 3 -4 0

Вывод на консоль: 3 1

Содержимое файла data.txt:

1

-2

3

-4

0

- Содержимое файла report.txt:

3

1

- Результат: тест пройден, программа работает корректно.

Тест 4: program.exe

Ввод: 0

Вывод на консоль: пусто

Содержимое файла data.txt:

0

- Содержимое файла report.txt:

Положительные числа в стеке:

пусто

- Результат: тест пройден, программа работает корректно.

**Выводы.**

В ходе выполнения лабораторной работы была разработана программа, которая позволяет: построить линейный односвязный список типа «стек» с заполнением его узлов, вводимыми с клавиатуры числами, вывод на экран в обратном порядке всех положительных чисел, хранимых в стеке.

# Список использованной литературы

1. Павловская Т.А., Чаевников В.В., Юрков Н.К. Программирование на языке С++. Электронное методическое пособие. СПб.: Издательство СПбГЭТУ «ЛЭТИ». 2014.
2. Практикум по процедурному программированию на языке C++ С. А. Ивановский, А. А. Лисс, В. П. Самойленко, О. М. Шолохова. Практикум по процедурному программированию на языке C++: учеб. пособие. СПб.: Издательство СПбГЭТУ«ЛЭТИ», 2016.
3. Справочник по стандартной библиотеке C++ (STL) // Справочный сайт learn.microsoft.com URL: https://learn.microsoft.com/ru-ru/cpp/standard-library/cpp-standard-library-reference?view=msvc-170 (дата обращения 11.02.2024).
4. Документация по языку C++ // Справочный сайт learn.microsoft.com URL: https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170 (дата обращения 11.02.2024).
5. Файловые потоки. Открытие и закрытие // Интернет платформа metanit.com URL: https://metanit.com/cpp/tutorial/8.2.php (дата обращения 11.02.2024).

# Приложение

## Код написанной программы

#include <iostream> // Подключение библиотеки для ввода-вывода

#include <windows.h> // Подключение библиотеки для работы с консолью

#include <fstream> // Подключение библиотеки для работы с файлами

using namespace std; // Использование пространства имен std

// Определение узла стека

struct Node {

int data; // Данные узла

Node\* next; // Указатель на следующий узел

};

// Определение класса стека

class Stack {

private:

Node\* top; // Вершина стека

public:

Stack() : top(nullptr) {} // Конструктор стека, инициализирует вершину nullptr

void push(int value) { // Метод для добавления элемента в стек

Node\* node = new Node(); // Создание нового узла

node->data = value; // Присваивание данных узлу

node->next = top; // Связывание узла с вершиной стека

top = node; // Обновление вершины стека

}

void pop() { // Метод для извлечения элемента из стека

if (top == nullptr) { // Если стек пуст

cout << "Стек пуст!\n"; // Выводим сообщение

return; // Завершаем метод

}

Node\* temp = top; // Временный указатель на вершину стека

top = top->next; // Перемещение вершины на следующий узел

delete temp; // Удаление узла

}

int peek() { // Метод для получения данных вершины стека

if (top == nullptr) { // Если стек пуст

cout << "Стек пуст!\n"; // Выводим сообщение

return -1; // Возвращаем -1

}

return top->data; // Возвращаем данные вершины

}

bool isEmpty() { // Метод для проверки на пустоту стека

return top == nullptr; // Стек пуст, если вершина равна nullptr

}

void transfer(Stack& from, Stack& to) { // Метод для переноса элементов из одного стека в другой

while (!from.isEmpty()) { // Пока исходный стек не пуст

to.push(from.peek()); // Добавляем в целевой стек данные вершины исходного стека

from.pop(); // Удаляем вершину исходного стека

}

}

void removeNegativeAndPrintPositive(Stack& stack) { // Метод для удаления отрицательных и вывода положительных элементов стека

Stack positive; // Создание стека для положительных элементов

while (!stack.isEmpty()) { // Пока исходный стек не пуст

int value = stack.peek(); // Сохранение данных вершины стека

stack.pop(); // Удаление вершины стека

if (value > 0) // Если данные положительные

positive.push(value); // Добавляем их в стек положительных элементов

}

transfer(positive, stack); // Переносим положительные элементы обратно в исходный стек

ofstream report("report.txt"); // Создаем объект для записи в файл

while (!stack.isEmpty()) { // Пока стек не пуст

cout << stack.peek() << "\n"; // Выводим число на консоль

report << stack.peek() << "\n"; // Записываем число в файл

stack.pop(); // Удаляем число из стека

}

report.close(); // Закрываем файл

}

};

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

{

SetConsoleCP(1251); // Установка кодировки консоли для ввода

SetConsoleOutputCP(1251); // Установка кодировки консоли для вывода

Stack stack; // Создание стека

int x = 1; // Переменная для ввода чисел

bool human = true; // Переменная для выбора интерфейса

if (argc > 1) { // Если есть аргумент командной строки

human = (argv[1][0] == 't'); // Присваиваем ему значение true или false

}

ofstream out("data.txt"); // Создаем объект для записи в файл

if (human) { // Если интерфейс для человека

cout << "Введите последовательность чисел. Для того, чтобы завершить ввод введите 0\n"; // Выводим подсказку

while (x != 0) { // Пока не введен 0

for (int i = 0; x != 0; i++) { // Для каждого введенного числа

cout << "a" << i << " = "; // Выводим индекс

cin >> x; // Считываем число

out << x << "\n"; // Записываем число в файл

stack.push(x); // Добавляем число в стек

}

}

out.close(); // Закрываем файл

cout << "Положительные числа в стеке: \n"; // Выводим подсказку

stack.removeNegativeAndPrintPositive(stack); // Удаляем отрицательные и выводим положительные числа

}

else // Если интерфейс для программы

{

while (x != 0) { // Пока не введен 0

cin >> x; // Считываем число с клавиатуры

out << x << "\n"; // Записываем число в файл

stack.push(x); // Добавляем число в стек

}

out.close(); // Закрываем файл

stack.removeNegativeAndPrintPositive(stack); // Удаляем отрицательные и выводим положительные числа

}

system("pause");

}