**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

**Тема: Лабораторная работа № 4: Динамические структуры данных**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Ермолаева В. А. |
| Преподаватель |  | Государкин Я. С. |

Санкт-Петербург

2024

## Цель работы

Ознакомиться со структурой данных стека и научиться применять базовые механизмы языка С++ для реализации стека в виде класса, а также методов для работы с ним, ввода и вывода данных программы и обработки возможных ошибок в процессе исполнения.

## Задание

Вариант 1.

1) Реализовать класс CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных int.

Объявление класса стека:

class CustomStack {

public:

// методы push, pop, size, empty, top + конструкторы, деструктор

private:

// поля класса, к которым не должно быть доступа извне

protected: // в этом блоке должен быть указатель на массив данных

int\* mData;

};

Перечень методов класса стека, которые должны быть реализованы:

* void push(int val) - добавляет новый элемент в стек
* void pop() - удаляет из стека последний элемент
* int top() - доступ к верхнему элементу
* size\_t size() - возвращает количество элементов в стеке
* bool empty() - проверяет отсутствие элементов в стеке
* extend(int n) - расширяет исходный массив на n ячеек

2) Обеспечить в программе считывание из потока stdin последовательности (не более 100 элементов) из чисел и арифметических операций (+, -, \*, / (деление нацело)) разделенных пробелом, которые программа должна интерпретировать и выполнить по следующим правилам:

* Если очередной элемент входной последовательности - число, то положить его в стек,
* Если очередной элемент - знак операции, то применить эту операцию над двумя верхними элементами стека, а результат положить обратно в стек (следует считать, что левый операнд выражения лежит в стеке глубже),
* Если входная последовательность закончилась, то вывести результат (число в стеке).

Если в процессе вычисления возникает ошибка:

* например вызов метода pop или top при пустом стеке (для операции в стеке не хватает аргументов),
* по завершении работы программы в стеке более одного элемента,

программа должна вывести "error" и завершиться.

## Выполнение работы

Описание функций:

* int main(): главная функция программы, возвращает 0 при успешном завершении. Обрабатывает ввод пользователя и потенциальные ошибки в процессе выполнения программы, работает со стеком и выводит результат.
* void push(int val): добавляет новый элемент в стек, проверяет стек на наличие свободного места и при необходимости выделяет больше памяти.
* void pop(): удаляет из стека последний элемент, если стек пустой, то выдает ошибку.
* int top(): доступ к верхнему элементу, если стек пустой, то выдает ошибку.
* size\_t size(): возвращает количество элементов в стеке.
* bool empty(): проверяет отсутствие элементов в стеке.
* extend(int n): расширяет исходный стек на n ячеек, если значение n отрицательное, выдает ошибку

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | 1 2 + 3 4 - 5 \* + | -2 | Выходные данные соответствуют ожиданиям. |
| 2. | 1 2 + 3 4 - 5 \* + 8 | error | Выходные данные соответствуют ожиданиям. |
| 3. | 1 2 + 3 X - 5 \* + | error | Выходные данные соответствуют ожиданиям. |

## Выводы

В ходе выполнения лабораторной работы были изучены и применены на практике принципы работы со стеком на языке С++. Освоены навыки, необходимые для реализации стека в виде класса, а также методов для работы с ним, ввода и вывода данных программы и обработки возможных ошибок в процессе исполнения.

# Приложение А Исходный код программы

#define BLOCK 10

class CustomStack {

private:

size\_t dataSize;

size\_t memSize;

protected:

int\* mData;

public:

CustomStack() {

this->mData = (int \*) malloc(BLOCK \* sizeof(int));

this->dataSize = 0;

this->memSize = BLOCK;

}

~ CustomStack() {

free(this->mData);

}

void push(int val) {

if (this->dataSize == this->memSize) {

this->memSize += BLOCK;

this->mData = (int \*) realloc(this->mData, this->memSize \* sizeof(int));

}

this->mData[this->dataSize++] = val;

}

void pop() {

if (this->dataSize <= 0) throw 0;

this->dataSize--;

}

int top() {

if (this->dataSize <= 0) throw 0;

return this->mData[this->dataSize - 1];

}

size\_t size() {

return this->dataSize;

}

bool empty() {

return this->dataSize == 0;

}

void extend(int n) {

if (n < 0) throw 0;

this->memSize += n;

this->mData = (int \*) realloc(this->mData, this->memSize \* sizeof(int));

}

};

int main() {

CustomStack stack;

char str[200];

fgets(str, 200, stdin);

char\* token = strtok(str, " ");

int error = 0;

while(token != NULL) {

try {

if (atoi(token) != 0 || strstr("0", token))

stack.push(atoi(token));

else if (strstr("+-\*/", token) && stack.size() > 1) {

int a = stack.top();

stack.pop();

int b = stack.top();

stack.pop();

if (strstr(token, "+"))

stack.push(a + b);

if (strstr(token, "-"))

stack.push(b - a);

if (strstr(token, "\*"))

stack.push(a \* b);

if (strstr(token, "/"))

stack.push(b / a);

}

else throw 0;

token = strtok(NULL, " ");

}

catch (int e) {

error = 1;

break;

}

}

if (stack.size() > 1) error = 1;

if(error) printf("error\n");

else printf("%d\n", stack.top());

return 0;

}