МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Программирование»
Тема: Динамические структуры данных

Студент гр. 3341 Гребенюк В.А.

Преподаватель Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Освоение работы с динамическими структурами данных в C++.

Задание

Моделирование стека.

Требуется написать программу, моделирующую работу стека на базе списка. Для этого необходимо:

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

Структура класса узла списка:

```
struct ListNode {
   ListNode* mNext;
   int mData;
};
```

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 голову
```

```
    ListNode* mHead;
};
```

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() - возвращает верхний элемент

size_t size() - возвращает количество элементов в стеке

bool empty() - проверяет отсутствие элементов в стеке

2) Обеспечить в программе считывание из потока stdin последовательности команд (каждая команда с новой строки), в зависимости от которых программа выполняет ту или иную операцию и выводит результат ее выполнения с новой строки.

Перечень команд, которые подаются на вход программе в stdin:

cmd_push n - добавляет целое число n в стек. Программа должна вывести "ok"

cmd_pop - удаляет из стека последний элемент и выводит его значение на экран

cmd_top - программа должна вывести верхний элемент стека на экран не удаляя его из стека

cmd_size - программа должна вывести количество элементов в стеке

cmd_exit - программа должна вывести "bye" и завершить работу

Если в процессе вычисления возникает ошибка (например вызов метода pop или top при пустом стеке), программа должна вывести "error" и завершиться.

Примечания:

Указатель на голову должен быть protected.

Подключать какие-то заголовочные файлы не требуется, всё необходимое подключено.

Предполагается, что пространство имен std уже доступно.

Использование ключевого слова using также не требуется.

Структуру ListNode реализовывать самому не надо, она уже реализована.

Выполнение работы

Разработанный программный код см. в приложении А.

Выводы

Работа с динамическими структурами на основе использующей программы освоена.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
class CustomStack {
public:
    void push(int val) {
        ListNode **cursor = this->last();
        if (*cursor != nullptr)
            cursor = &(*cursor)->mNext;
        *cursor = new ListNode();
        (*cursor)->mData = val;
        this->len++;
    }
    void pop() {
        if (!this->len) return;
        ListNode **nodeToDelete = this->last();
        if (this->len > 1) {
            ListNode **secondToLast = this->last(-1);
            delete *nodeToDelete;
            (*secondToLast)->mNext = nullptr;
        } else {
            delete *nodeToDelete;
            this->mHead = nullptr;
        }
        this->len--;
    }
    int top() {
        if (this->len == 0)
            throw exception();
        return (*this->last())->mData;
    }
    size_t size() const {
        return this->len;
    }

    bool empty() {
        return this->mHead != nullptr;
    }

private:
    ListNode **last(int offset = 0) {
        ListNode **cursor = &this->mHead;
        for (auto i = 1; i < this->len + offset; i++)
            cursor = &(*cursor)->mNext;
        return cursor;
    }
    size_t len = 0;

protected:
    ListNode *mHead = nullptr;
};
```

```
void run_commands(CustomStack &stack) {
    string buffer;
    try {
        while (true) {
            cin >> buffer;

            if (buffer.compare("cmd_push") == 0) {
                int val;
                cin >> val;
                stack.push(val);
                cout << "ok\n";
            } else if (buffer.compare("cmd_pop") == 0) {
                cout << stack.top() << '\n';
                stack.pop();
            } else if (buffer.compare("cmd_top") == 0) {
                cout << stack.top() << '\n';
            } else if (buffer.compare("cmd_size") == 0) {
                cout << stack.size() << '\n';
            } else if (buffer.compare("cmd_exit") == 0) {
                cout << "bye" << endl;
                break;
            } else {
                throw exception();
                break;
            }
        }
    } catch (const exception &e) {
        cout << "error\n"
             << endl;
    }
}

int main() {
    CustomStack stack = CustomStack();
    run_commands(stack);
    return 0;
}
```