Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 7

по курсу «Языки и методы программирования»

«Разработка простейшего класса на C++»

Студент группы ИУ9-22Б Федуков А. А.

Преподаватель Посевин Д. П.

24 апреля 2024 г.

Цель работы

Целью данной работы является изучение базовых объектно-ориентированных возможностей языка C++.

Задание

Выполнение лабораторной работы заключается в составлении на языке C++ программы, состоящей из трёх файлов: • заголовочный файл declaration.h с объявлением одного из классов, приведённых в таблицах 1 – 16; • файл implementation.cpp с определениями методов класса; • файл main.cpp, содержащий функцию main и, возможно, вспомогательные функции для проверки работоспособности класса. Реализация класса не должна опираться на стандартные контейнерные классы C++, то есть внутреннее состояние объектов класса должно быть реализовано через обычные массивы. Соответственно, в классе обязательно требуется реализовать: • конструктор копий; • деструктор (должен быть объявлен виртуальным); • операцию присваивания. Проверку работоспособности класса требуется организовать в функции main, размещённой в файле main.cpp. Проверка должна включать в себя: • создание объекта класса в автоматической памяти; • передачу объекта класса по значению в функцию; • присваивание объекта класса переменной.

Задание 1

Вариант 9

Стековая машина, оперирующая вещественными числами, с операциями:

1. получение количества чисел в стеке;
2. добавление вещественного числа на стек;
3. получение ссылки на i-ое число, считая от вершины стека;
4. сложение, умножение и вычитание двух чисел на вершине стека (числа удаляются со стека, результат добавляется в стек).

Реализация

Я описал класс в файле заголовков declaration.h

Реализовал его в implementation.cpp

Сгенирировал его экземпляры и проверил работоспособность необходимых функций уже в main.cpp

Код

Листинг 1: Файл declaration.h

```
1  #ifndef DECLARATION_H
2  #define DECLARATION_H
3  #include <iostream>
4
5  class Stack
6  {
7  private:
8      int capacity;
9      int size;
10     float *data;
11
12 public:
13     Stack(); // Конструктор
14     Stack(const Stack& stk); // Конструктор копирования
15     virtual ~Stack(); // Деструктор
16     Stack& operator=(const Stack& counter); // Оператор присваивания
17
18     int getSize();
19     void add(float x);
20     void doubleCapacity();
21     float* get(int i);
22     float top();
23     void addition();
24     void subtraction();
25     void multiplication();
26 };
27
28
29 #endif
```

Листинг 2: Файл implementation.cpp

```
#define INITIALSTACKCAPACITY 5
#include "declaration.h"
Stack::Stack(){
    Stack::capacity = INITIALSTACKCAPACITY;
    Stack::data = new float[INITIALSTACKCAPACITY];
    Stack::size = 0;
}

Stack::~Stack(){
    delete[] data;
    std::cout << "Stack was deleted" << std::endl;
}

Stack& Stack::operator=(const Stack& stk)
{
    float* data_new = new float[stk.capacity];
    for (size_t i = 0; i < stk.size; i++)
    {
        data_new[i] = stk.data[stk.size - i - 1];
    }
    data = data_new;
    size = stk.size;
    capacity = stk.capacity;

    return *this;
}

Stack::Stack(const Stack &stk){
    float* data_new = new float[stk.capacity];
    for (size_t i = 0; i < stk.size; i++)
    {
        data_new[i] = stk.data[i];
    }
    data = data_new;
    size = stk.size;
    capacity = stk.capacity;
}

int Stack::getSize(){
    return this->size;
}

void Stack::doubleCapacity(){
    float * data_new = new float[this->capacity*2];
    for (size_t i = 0; i < this->size; i++)
```

```
46      {
47          data_new[i] = this->data[i];
48      }
49      delete[] this->data;
50      this->data = data_new;
51      this->capacity *= 2;
52  }
53
54  void Stack::add(float x){
55      if (capacity <= size + 1)
56          doubleCapacity();
57
58      data[size] = x;
59      size += 1;
60
61  }
62
63  float Stack::top() {
64      if (size == 0)
65      {
66          std::cerr << "Stack is empty" << std::endl;
67          return 0;
68      }
69      size -= 1;
70      return data[size];
71  }
72
73  void Stack::addition(){
74      Stack::add(Stack::top() + Stack::top());
75  }
76
77  void Stack::subtraction(){
78      Stack::add(-Stack::top() + Stack::top());
79  }
80  void Stack::multiplication(){
81      Stack::add(Stack::top() * Stack::top());
82
83  }
84
85  float* Stack::get(int i){
86      if (size < i)
87      {
88          std::cerr << "Stack has not enough data" << std::endl;
89          return NULL;
90      }
91      return &data[size - 1 - i];
```

```
92 }
```

Листинг 3: Файл main.cpp

```
1  #include "implementation.cpp"
2
3  void print_stack(Stack& stk, char n){
4      int i = stk.getSize();
5      std::cout << "Stk" << n << ": ";
6      for (size_t i = 0; i < stk.getSize(); i++)
7      {
8          std::cout << *stk.get(i) << " ";
9      }
10     std::cout << std::endl;
11
12 }
13
14 int main()
15 {
16     Stack stk, stk2;
17     stk.add(10);
18     stk.add(9);
19     stk.add(8);
20     print_stack(stk, '1');
21     print_stack(stk2, '2');
22     stk2 = stk; // Assigment (purposely reversed)
23     std::cout << "Assigment stk2 = stk" << std::endl;
24     print_stack(stk2, '2');
25     auto stk3 = stk; // Copy
26     std::cout << "Copy stk to stk3" << std::endl;
27     std::cout << "stk top: " << stk.top() << std::endl;
28     stk2.multiplication();
29     std::cout << "stk2 top: " << stk2.top() << std::endl;
30     std::cout << "stk2 top: " << stk2.top() << std::endl;
31     print_stack(stk, '1');
32     print_stack(stk2, '2');
33     print_stack(stk3, '3');
34     return 0;
35 }
```

Вывод программы

Программа создала экземляры класса и протестировала все заявленные операции

Листинг 4: Вывод программы

```
Stk1: 8 9 10
Stk2:
Assigment stk2 = stk
Stk2: 10 9 8
Copy stk to stk3
stk top: 8
stk2 top: 90
stk2 top: 8
Stk1: 9 10
Stk2:
Stk3: 8 9 10
Stack was deleted
Stack was deleted
Stack was deleted
```

Задание 2

Вариант 19

Абсолютный путь к каталогу в файловой системе UNIX с операциями:

1. получение количества каталогов в пути;
2. получение ссылки на i-тый каталог в пути, считая от корня;
3. добавление имени каталога в конец пути;
4. получение количества файлов в каталоге (для этого требуется разобраться, как получить оглавление каталога).

Реализация

Я описал класс в файле заголовков declaration.h
Реализовал его в implementation.cpp
Сгенирировал его экземпляры и проверил работоспособность необходимых функций уже в main.cpp

Код

Листинг 5: Файл declaration.h

```
#ifndef DECLARATION_H
#define DECLARATION_H
#include <iostream>
#include <filesystem>
#include <string>
#include <algorithm>
class PathToDir
{
    friend PathToDir addPath(PathToDir &Pth, std::string path);
private:
    std::string path;
    int dirsNumber;
    std::string *dirs;

public:
    PathToDir(std::string path); // Конструктор
    PathToDir(const PathToDir& pth); // Конструктор копирования
    virtual ~PathToDir(); // Деструктор
    PathToDir& operator=(const PathToDir& counter); // Оператор присваив
    ания

    int getNumberOfDirsInPath(); // получение количества каталогов в пут
    и;
    std::string& getDir(int k); // получение ссылки на i-тый каталог в п
    ути, считая от корня;
    void addDir(std::string dirName); // добавление имени каталога в кон
    ец пути;
    int getNumberOfFiles(); // получение количества файлов в каталоге (д
    ля этого требуется разобраться, как получить оглавление каталога).

};


#endif
```

Листинг 6: Файл implementation.cpp

```
#include "declaration.h"
PathToDir::PathToDir(std::string pth){
    if (pth.back() != '/')
    {
        pth += '/';
    }
```

```
7       path = pth;
8       dirsNumber = std::count(path.begin(), path.end(), '/');
9       dirs = new std::string[dirsNumber];
10      dirs[0] = "/";
11      pth = pth.substr(1);
12      size_t pos = 0;
13      int i = 1;
14      while ((pos = pth.find('/')) != std::string::npos) {
15          dirs[i] = pth.substr(0, pos);
16          pth.erase(0, pos + 1);
17          i += 1;
18      }
19  }
20  PathToDir::PathToDir(const PathToDir &Pth){
21      path = Pth.path;
22      dirsNumber = Pth.dirsNumber;
23      std::string *dirs_new = new std::string[Pth.dirsNumber];
24      for (size_t i = 0; i < dirsNumber; i++)
25      {
26          dirs_new[i] = Pth.dirs[i];
27      }
28      dirs = dirs_new;
29  }
30  PathToDir::~PathToDir(){
31      delete[] dirs;
32      std::cout << "PathToDir was deleted" << std::endl;
33  }
34
35  PathToDir& PathToDir::operator=(const PathToDir& Pth)
36  {
37      path = Pth.path;
38      dirsNumber = Pth.dirsNumber;
39      std::string *dirs_new = new std::string[Pth.dirsNumber];
40      for (size_t i = 0; i < dirsNumber; i++)
41      {
42          dirs_new[i] = Pth.dirs[i];
43      }
44      dirs = dirs_new;
45      return *this;
46  }
47
48  int PathToDir::getNumberOfDirsInPath(){
49      return dirsNumber;
50  }
51
52  std::string& PathToDir::getDir(int k){
```

```
53      return *(dirs + k - 1);
54  }
55
56  void PathToDir::addDir(std::string dirName){
57      *this = PathToDir(path + dirName + "/");
58  }
59
60  int PathToDir::getNumberOfFiles(){
61      int counter = 0;
62      for (const auto & entry : std::filesystem::directory_iterator(path))
        {
63          if (!entry.is_directory()){
64              counter += 1;
65          }
66      }
67      return counter;
68  }
```

Листинг 7: Файл main.cpp

```
1  #include "implementation.cpp"
2
3  #include <string>
4  #include <iostream>
5  #include <filesystem>
6
7  PathToDir addPath(PathToDir &Pth, std::string path){
8      return PathToDir(Pth.path + path);
9  }
10
11 int main()
12 {
13     PathToDir Pth("/home/chinalap/");
14     std::cout << "Number of dirs: " << Pth.getNumberOfDirsInPath() <<
       std::endl;
15     Pth.addDir("Документы");
16     std::cout << "The forth dir: " << Pth.getDir(4) << std::endl;
17     PathToDir Pth2 = Pth;
18     std::cout << "Number of dirs: " << Pth2.getNumberOfDirsInPath() <<
       std::endl;
19     Pth = addPath(Pth2, "YIMP_labs/lab7/files/code/task2");
20     std::cout << "Number of files: " << Pth.getNumberOfFiles() << std::
       endl;
21
22 }
```

Вывод программы

Программа создала экземляры класса и протестировала все заявленные операции

Листинг 8: Вывод программы

```
1 Number of dirs: 3
2 PathToDir was deleted
3 The forth dir: Документы
4 Number of dirs: 4
5 PathToDir was deleted
6 Number of files: 4
7 PathToDir was deleted
8 PathToDir was deleted
```

Вывод

Выполняя эту лабораторную работу я попробовал применить знания об ООП, полученные ранее, во время работы с Java. Я описал некоторые классы и их некоторый фукнкционал. Мне показалось, что на C++ это делать сложнее, хотя и выходит лаконичнее.