**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»**

**Факультет информатики**

**Кафедра технической кибернетики**

**Отчет по лабораторной работе №5**

**по курсу «Языки программирования»**

Выполнил: Борисов Д. С.

Проверил: Головастиков Н.В.

Группа: 6110

САМАРА 2017

**Задание**

**Основное задание на ЛР:**

Реализовать на C++ консольное приложение, предназначенное для работы с массивом экземпляров класса. **Входные данные:** количество элементов массива, экземпляры класса. **Выходные данные:** массив экземпляров класса и массив результатов работы метода по обработке данных.

**Особенности:**

**-** у класса должны быть перегружены операторы взятия из потока и помещения в поток;

- должна быть создана функция-член класса, реализующая обработку данных экземпляра класса согласно варианту;

- должна быть реализована возможность выбора чтения из файла или с консоли;

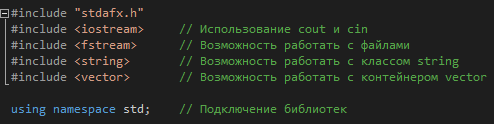
- должна быть реализована возможность выбора записи в файла или вывода в консоль;.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Данные о скорости | *Номер измерения Скорость Единица измерения* | 2 60 km/h | Вывод скорости в км/ч и м/с |

**Основная часть**

Для решения поставленной задачи воспользуемся программой Microsoft Visual Studio, используя язык программирования С++.

Для успешного выполнения задания необходимо подключить необходимые библиотеки, в которых хранятся команды для работы.



*Рис. 1. Подключение библиотек*

Теперь необходимо создать класс Speed, имеющий три переменных (номер измерения –“int”, значение скорости – “double”, единица измерения – “string”). Эти три переменных создаются в приватной части класса, а все его методы, перегрузки операторов для него и так далее в публичной части.

**В публичной части находятся:**



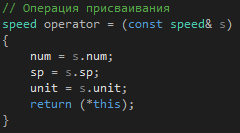
*Рис. 1. Создание пустого объекта класса*



*Рис. 2. Создание заполненного класса.*

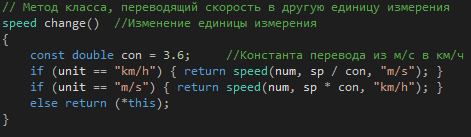


*Рис. 3. Операция копирования для класса*



*Рис. 4. Операция присваивания для класса*

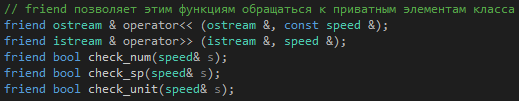
Перегруженные операторы: +, +=, -, -=, \*, /, <, >, ==.



*Рис. 5. Метод класса, переводящий скорость в другую единицу измерения*



*Рис. 6. Деструктор для удаления класса*



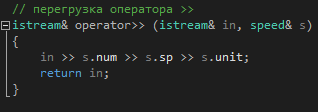
*Рис. 7. Список «дружелюбных» функций к классу*

На этом **класс заканчивается**, и далее идут уже стороние функции и две перегрузки операторов ввода и вывода для класса.

Сначала идут четыре функции проверки значений класса. Первая функция **(check)** вызывает три последующие булевы функции и возващает true, только если все три вызванные функции возвращают значение true. Вторая функция **(check\_num)** проверяет первый элемент класса (возвращает true, если он больше нуля, иначе выводит на экран сообщение о ошибке и возвращает false). Третья функция **(check\_sp)** проверяет значение скорости на положительность (возвращает true, если оно неотрицательное, иначе выводит на экран сообщение о ошибке и возвращает false). Четвёртая функция **(check\_unit)** проверяет правильность единицы измерения скорости (возвращает true, если это “km/h” или “m/s”, иначе выводит на экран сообщение о ошибке и возвращает false).

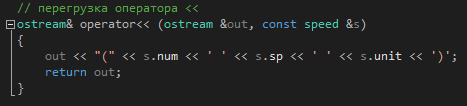
Теперь идут две перегрузки для класса: занесения в поток и чтение из потока.

**Чтение из потока:** мы ссылаемся на поток и на объект класса; заносим по-порядку номер, скорость, единицу измерения.



*Рис. 8. Чтение из потока*

**Занесение в поток:** мы ссылаемся на поток и на объект класса; заносим в поток: скобку, номер, пробел, скорость, пробел, единицу измерения, скобку.



*Рис. 9. Занесение в поток*

Три следующие функции связаны друг с другом. Первая функция **(output)** предоставляет выбор места куда будет заносится результат (в консоль или в файл), поэтому внутри него находится бесконечный цикл while, выйти из которого можно лишь правильно ответив на вопрос, иначе выйдет сообщение о неправильном ответе и просьба ввести его заново, тем самым цикл перейдёт в следующую итерацию и будет дана ещё одна попытка ввести правильный ответ. Если пользователь решил занести результаты в консоль, то сработает вторая функция **(coutput)**, которая выводит строчку со старыми значениями из вектора “sp1”, и строчку с новыми значениями из вектора “sp2”. Если пользователь решил занести результаты в файл, то сработает третья функция **(foutput),** создающая файл (или заменяющая на новый), в который будут занесены две строчки со старыми значениями и с новыми значениями.

Следом идёт простенькая функция **cont,** которая предлагает пользователю продолжить работу, возвращающая true, если пользоваетель решил продолжить и false, если он решил прекратить работу.

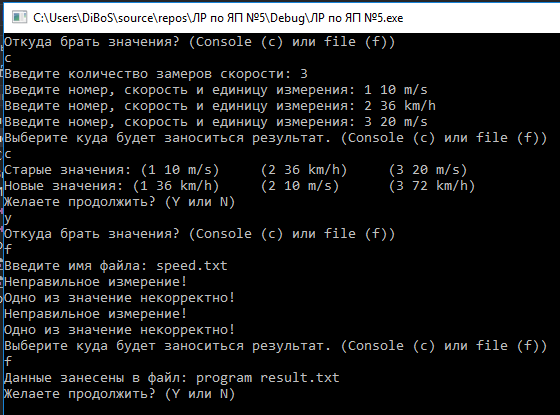
После неё идут две “параллельные”функции: **console** и **file.**

**Функция console:** Пользователь вводит количество замеров скорости. Начинается цикл for с количеством итераций равный введённому числу. В каждой итерации пользователь заносит данные в объект класса Speed под названием old, и если после инициализации класса, функция check от этого класса возвращает false, то счётчик цикла i уменьшается на единицу и мы переходим к новой итерации, тем самым пользователь должен будет заново ввести данные, пока check не вернёт true. Если класс инициализирован правильно, то он заносится в вектор sp1, а в вектор sp2 заносится его изменённая версия с помощью метода change. В итоге каждый вектор будет содержать количество элементов равный количеству замеров скорости. После цикла for идёт вызов функции output, где даётся выбор места занесения результатов. После чего эти два вектора очищаются, чтобы с ними можно было в дальнейшем работать, если пользователь решит продолжить.

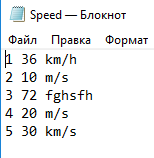
**Функция file:** Она выполняет ту же самую цель, что и функция console, с небольшими изменениями. В file значения берутся из файла, имя которого пользователь должен будет ввести. После проверки на его открытие, через цикл while каждое значение из файла поочерёдно берётся и заносится в два вектора (по тому же принципу). Затем вызывается функция output, после которой эти два вектора очищаются.

Ну и наконец, сердце программы – **функция main.** В ней создаются два вектора sp1 и sp2, после чего пользователь должен выбрать откуда брать значения (из консоли или из файла).

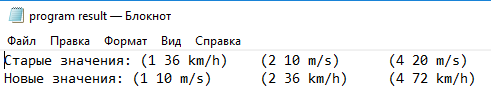
Если пользователь решил брать значения из консоли, т.е. самостоятельно вводить данные, то вызывается функция console. Если пользователь решил брать значения из файла, то вызывается функция file. После одной из них программа закрывается (если функция cont вернула false) или начинает все действия заново (если функция cont вернула true).



*Рис. 10. Листинг программы*



*Рис. 11. Файл Speed.txt*



*Рис. 12. Файл program result.txt*

Заключение

Цель лабораторной работы - написать программу, которая переводит объект самостоятельно созданного класса со значениями скорости в другую единицу измерения, достигнута. В ходе выполнения лабораторной работы я изучил принципы работы класса и его методов, приобрел навыки по созданию собственных классов.

Литература

1. Безус Е.А. Козлова Е.С. Лабораторные работы по курсу «Языки программирования» (язык программирования C++): Методические указания / Сост. Е.А. Безус, Е.С. Козлова; Самара, 2016. 36 с.
2. Страуструп Б. Язык программирования С++ / Б. Страуструп; М:Бином, 2011. 1136 с.

Приложение

#include "stdafx.h"

#include <iostream> // Использование cout и cin

#include <fstream> // Возможность работать с файлами

#include <string> // Возможность работать с классом string

#include <vector> // Возможность работать с контейнером vector

using namespace std; // Подключение библиотек

//Объявление класса Speed

class Speed

{

private:

int num;

double sp;

string unit;

public:

Speed() // Создание пустого класса

{

};

// Создание заполненного класса

Speed(int n, double s, string un) : num(n), sp(s), unit(un) { }

// Операция копирования

Speed(const Speed& s) : num(s.num), sp(s.sp), unit(s.unit) { }

// Прочие перегрузки

// Операция присваивания

Speed operator = (const Speed& s)

{

num = s.num;

sp = s.sp;

unit = s.unit;

return (\*this);

}

// Арифметические операторы

Speed operator += (const double& d) // Оператор суммы

{

return Speed(num, sp += d, unit);

}

Speed operator + (const double& d) // Оператор суммы

{

return Speed(num, sp + d, unit);

}

Speed operator -= (const double& d) // Оператор разности

{

return Speed(num, sp -= d, unit);

}

Speed operator - (const double& d) // Оператор разности

{

return Speed(num, sp - d, unit);

}

Speed operator \* (const double& d) // Оператор умножения

{

return Speed(num, sp \* d, unit);

}

Speed operator / (const double& d) // Оператор деления

{

if (d != 0) { return Speed(num, sp / d, unit); }

else return (\*this);

}

// Операторы сравнения

bool operator < (const Speed& s)

{

if (unit == s.unit) { return sp < s.sp; }

if (unit == "km/h" && s.unit == "m/s") { return sp < (s.sp \* 3.6); }

if (unit == "m/s" && s.unit == "km/h") { return (sp \* 3.6) < s.sp; }

}

bool operator > (const Speed& s)

{

if (unit == s.unit) { return s.sp < sp; }

if (unit == "km/h" && s.unit == "m/s") { return (s.sp \* 3.6) < sp; }

if (unit == "m/s" && s.unit == "km/h") { return s.sp < (sp \* 3.6); }

}

bool operator == (const Speed& s)

{

if (unit == s.unit) { return !(sp < s.sp) && !(s.sp < sp); }

if (unit == "km/h" && s.unit == "m/s") { return !(sp < (s.sp \* 3.6)) && !((s.sp \* 3.6) < sp); }

if (unit == "m/s" && s.unit == "km/h") { return !((sp \* 3.6) < s.sp) && !(s.sp < (sp \* 3.6)); }

}

// Метод класса, переводящий скорость в другую единицу измерения

Speed change() //Изменение единицы измерения

{

const double con = 3.6; //Константа перевода из м/с в км/ч

if (unit == "km/h") { return Speed(num, sp / con, "m/s"); }

if (unit == "m/s") { return Speed(num, sp \* con, "km/h"); }

else return (\*this);

}

~Speed() // Деструктор для удаления класса

{

}

// friend позволяет этим функциям обращаться к приватным элементам класса

friend ostream & operator<< (ostream &, const Speed &);

friend istream & operator>> (istream &, Speed &);

friend bool check\_num(Speed& s);

friend bool check\_sp(Speed& s);

friend bool check\_unit(Speed& s);

};

// Проверки значений класса

bool check(Speed& s)

{

return check\_num(s) && check\_sp(s) && check\_unit(s);

}

bool check\_num(Speed& s)

{

if (s.num < 1) {

cout << "Неправильный номер!\n";

return false;

}

else return true;

}

bool check\_sp(Speed& s)

{

if (s.sp < 0) {

cout << "Неправильная скорость!\n";

return false;

}

else return true;

}

bool check\_unit(Speed& s)

{

if ((s.unit != "km/h") && (s.unit != "m/s")) {

cout << "Неправильное измерение!\n";

return false;

}

else return true;

}

// перегрузка оператора >>

istream& operator>> (istream& in, Speed& s)

{

in >> s.num >> s.sp >> s.unit;

return in;

}

// перегрузка оператора <<

ostream& operator<< (ostream &out, const Speed &s)

{

out << "(" << s.num << ' ' << s.sp << ' ' << s.unit << ')';

return out;

}

// Вывод двух массивов на консоль

void coutput(const vector<Speed>& sp1, const vector<Speed>& sp2)

{

cout << "Старые значения: ";

for (Speed i : sp1) { cout << i << '\t'; }

cout << endl << "Новые значения: ";

for (Speed i : sp2) { cout << i << '\t'; }

cout << endl;

}

// Вывод двух массивов в файл

void foutput(const vector<Speed>& sp1, const vector<Speed>& sp2)

{

ofstream fout("program result.txt");

fout << "Старые значения: ";

for (Speed i : sp1) { fout << i << '\t'; }

fout << endl << "Новые значения: ";

for (Speed i : sp2) { fout << i << '\t'; }

cout << "Данные занесены в файл: program result.txt\n";

}

// Выбор размещения результатов

void output(const vector<Speed>& sp1, const vector<Speed>& sp2)

{

string f\_c;

cout << "Выберите куда будет заноситься результат. (Console (с) или file (f))\n";

while (getline(cin, f\_c)) {

if (f\_c == "") { continue; }

if (f\_c == "console" || f\_c == "Console" || f\_c == "c" || f\_c == "C") {

coutput(sp1, sp2);

return;

}

else if (f\_c == "file" || f\_c == "File" || f\_c == "f" || f\_c == "F") {

foutput(sp1, sp2);

return;

}

else cout << "Некорректный ответ! Введите его заново: ";

}

}

// Предложение продолжить работу

bool cont()

{

string y\_n;

cout << "Желаете продолжить? (Y или N)\n";

while (getline(cin, y\_n)) {

if (y\_n == "") { continue; }

if (y\_n == "Y" || y\_n == "y") { return true; }

else if (y\_n == "N" || y\_n == "n") { return false; }

else cout << "Некорректный ответ! Введите его заново: ";

}

}

// Работа с консолью

void console(vector<Speed>& sp1, vector<Speed>& sp2)

{

int size;

cout << "Введите количество замеров скорости: ";

cin >> size;

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

Speed old;

cout << "Введите номер, скорость и единицу измерения: ";

cin >> old;

if (!check(old)) {

--i;

continue;

}

sp1.push\_back(old);

sp2.push\_back(old.change());

}

output(sp1, sp2);

sp1.clear();

sp2.clear();

return;

}

// Работа с файлом

void file(vector<Speed>& sp1, vector<Speed>& sp2)

{

string file;

cout << "Введите имя файла: ";

while (getline(cin, file)) {

ifstream fin(file);

if (!fin.is\_open()) { cout << "Файл не найден. Введите его имя заново: "; }

else {

Speed old;

while (fin >> old) {

if (check(old)) {

sp1.push\_back(old);

sp2.push\_back(old.change());

}

else cout << "Одно из значение некорректно!\n";

}

output(sp1, sp2);

sp1.clear();

sp2.clear();

return;

}

}

}

// Сердце программы

int main()

{

setlocale(LC\_ALL, "rus");

string c\_f;

vector<Speed> sp1;

vector<Speed> sp2;

cout << "Откуда брать значения? (Console (с) или file (f))\n";

while (getline(cin, c\_f)) {

if (c\_f == "") { continue; }

if (c\_f == "console" || c\_f == "Console" || c\_f == "c" || c\_f == "C") {

console(sp1, sp2);

if (!cont()) { return 0; }

else cout << "Откуда брать значения? (Console (с) или file (f))\n";

}

else if (c\_f == "file" || c\_f == "File" || c\_f == "f" || c\_f == "F") {

file(sp1, sp2);

if (!cont()) { return 0; }

else cout << "Откуда брать значения? (Console (с) или file (f))\n";

}

else cout << "Некорректный ответ! Введите его заново: ";

}

}