1. Постановка задачи

2. Использование стандартной библиотеки языка C++

3. Описание иерархии объектов и методов объектов

4. Описание алгоритма основной программы

5. Тексты модуля, программы

6. Результат тестирования на различных данных

7. Список использованных источников

1. Постановка задачи

Разработать программное обеспечение для работы с информацией об учебных заведениях среднего общего образования (школы, лицеи, гимназии), структуру данных и метод получения различной информации.

1. Использование стандартной библиотека языка C++

При разработке программ на языке C++ весьма ускоряет процесс и облегчает работу использование многолетних наработок, объединённых в одну большую библиотеку языка C++, называемую Стандартная библиотека языка программирования C++. В языке программирования C++ термин Стандартная Библиотека означает коллекцию классов и функций, написанных на базовом языке. Стандартная Библиотека поддерживает несколько основных контейнеров, функций для работы с этими контейнерами, объектов-функции, основных типов строк и потоков (включая интерактивный и файловый ввод-вывод), поддержку некоторых языковых особенностей, и часто используемые функции для выполнения таких задач, как, например, нахождение квадратного корня числа. Стандартная Библиотека языка C++ также включает в себя спецификации стандарта ISO C90 стандартной библиотеки языка Си. Функциональные особенности Стандартной Библиотеки объявляются внутри пространства имен std.

Стандартная библиотека шаблонов (STL) — подмножество стандартной библиотеки C++ и содержит контейнеры, алгоритмы, итераторы, объекты-функции и т. д. Заголовочные файлы стандартной библиотеки C++ не имеют расширения «.h».

Для управления наборами объектов в стандартной библиотеке C++ определены контейнеры. Контейнер представляет коллекцию объектов определенного типа. Последовательный же контейнер (sequential container) позволяет контролировать порядок, в котором элементы располагаются в коллекции, и управлять доступом к этим элементам.

Типы последовательных контейнеров:

* **vector**: массив переменного размера.

Поддерживает произвольный доступ к любому элементу в контейнере.

Обеспечивает добавление и удаление элементов из любого места контейнера.

* **deque**: двусторонняя очередь.

Поддерживает произвольный доступ к любому элементу в контейнере.

Обеспечивает удаление и добавление элементов в начале и в конце контейнера.

* **list**: двухсвязный список

Поддерживает только последовательный двухнаправленный доступ к элементам.

Обеспечивает удаление и добавление элементов в начале и в конце контейнера.

* **forward\_list**: односвязный список.

Поддерживает только однонаправленный последовательный доступ к элементам.

Обеспечивает удаление и добавление элементов в начале и в конце контейнера.

* **array**: массив фиксированного размера.

Поддерживает произвольный доступ к любому элементу в контейнере.

Добавлять или удалять элементы из контейнера нельзя.

* **string**: представляет контейнер, аналогичный вектору, который состоит из символов, то есть строку

Таким образом, стандартная библиотека C++ по умолчанию содержит ряд контейнеров, которые представляют определенные структуры данных. Все они имеют как некоторые общие, так и специфические возможности. За исключением класса array все они поддерживают добавление и удаление элементов. Основное различие между ними состоит в том, как они обеспечивают добавление и удаление элементов, а также доступ к элементам в контейнере. И в зависимости от ситуации и потребностей можно использовать тот или иной тип контейнеров.

Для использования определенного контейнера в программу необходимо добавить соответствующий заголовочный файл, который, как правило, называется по имени класса контейнера.

Кроме того, при определении переменной контейнера необходимо установить ограничение на тип элементов, которые будет содержать контейнер. Тип элементов передается в угловых скобках. Например, определение переменной класса vector, которая хранит числовые данные типа int:

std::vector<int> v1;

Для дальнейшей работы из стандартной библиотеки весьма полезны конструкции типа vector, string, потоковый ввод-вывод.

Класс string позволяет работать со строками, предоставляя большое количество функций, реализованных как непосредственно методы класса, таки дополнительно для него. Потоки ввода вывода позволяют читать данные из файла и писать данные в файл используя различные подходы.

ifstream FRead(FName);

int bufsize = 256;

char \*buff = new char[bufsize]; // буфер промежуточного хранения считываемого из файла текста

if (FRead.is\_open()) // если файл открыт

{

FRead.getline(buff, bufsize) // Читаем строку из файла

}

Возможен также другой вариант

if (FRead.is\_open()) // если файл открыт

{

FRead >> str; // Читаем строку из файла

}

Контейнер vector std::vector<T> v1;

В качестве T может быть любой тип данных, в том числе и сам вектор ещё раз, таким образом можно делать многомерные массивы переменной длины.

1. Описание иерархии объектов и методов объектов
   1. Класс enum SchoolKind

Для указания вида учебного заведения создан тип-перечисление, который содержит в себе элементы, которые используются для указания видов учебных заведений.

enum SchoolKind

{

school=0, lyceum, gymnasium

};

* 1. Класс Info

Этот класс был разработан для хранения информации о результатах выпуска по одному году.

class Info

{

public:

int Year; // Год

int CountAll; // Количество выпускников

int CountGoldMedal; // Количество золотых медалей

int CountSilverMedal; // Количество серебряных медалей

int CountHigher; // Количество поступивших в высшие заведения

int CountMiddle; // Количество поступивших в средние заведения

};

* 1. Класс School

Данный класс работает с информацией об учебном заведении. Хранится индивидуальный код заведения, его вид, массив данных о выпусках.

class School

{

public:

School();

~School();

string code; // Код заведения

SchoolKind kind; // Вид заведения

vector<Info> YearInfo; // Информация о выпусках по годам

void GetData(string str); // Получение данных из строки

void ToCSV(vector<string> &v); // Формирование массива строк по массиву данных

int countGoldMedals(int year); // Количество золотых медалистов, параметр - год

int countAll(int year); // Количество выпускников, параметр - год

int countHigher(int year); // Количество выпускников, поступивших в ВУЗы, параметр - год

//Person director;

};

В классе реализованы следующие методы

void GetData(string str);

Получение данных из строки в формате хранения данных, поля разделены символом”:”.

void ToCSV(vector<string> &v);

Формирование массива строк по массиву данных в формат хранения данных, поля разделены символом”:”.

int countGoldMedals(int year);

Количество золотых медалистов, параметр - год

int countAll(int year);

Количество выпускников, параметр - год

int countHigher(int year);

Количество выпускников, поступивших в ВУЗы, параметр - год

* 1. Класс Schools

Разработан для обработки данных о всех учебных заведениях. Имеет публичное поле типа вектор с элементами типа класс School.

class Schools

{

public:

Schools();

~Schools();

bool ReadFromFile(const string FName);

bool WriteToFile(const string FName);

int countGoldMedals(int year);

int countAll(string code, int year);

int countHigher(string code, int year);

private:

vector<School> schools;

};

Реализованы функции чтения данных из файла и записи в файл в специальном формате. Формат записи текстовый, разделитель полей точка с запятой(формат csv).

Реализованы методы

- нахождения количества золотых медалистов во всех учебных заведениях в конкретном году

- нахождения количества выпускников в конкретном учебном заведении в указанном году

- нахождения количества выпускников, поступивших в ВУЗы, в конкретном учебном заведении в указанном году

* 1. Схема взаимосвязи классов

Класс Schools содержит поле, являющееся массивом(вектором) из элементов класса School и поле, являющееся объектом типа SchoolKind.

Класс School содержит поле, являющееся массивом(вектором) из элементов класса Info.

1. Основная программа

В правильно спроектированной программе на основе ООП все основные действия, как правило, реализованы в классах. В основной программе создаётся объект класса и у него вызываются методы, реализующие необходимые функции. Если со временем возникает задача нахождения какой-либо информации, осуществления расчётов, то более целесообразно реализовать это в классе, чем каждый раз писать эти функции в разных местах большого проекта. В связи с этим основная программа работы с классами состоит из небольшого количества операторов, демонстрирующих работу методов. В программе SchoolsTest.cpp реализованы следующие действия с классом Schools

- создание объекта класса

- вызов метода чтения данных из файла

- запрос ввода года

- по заданному коду учебного заведения и году выпуска найдены данные об общем количестве выпускников, о количестве поступивших в ВУЗы

- посчитан процент поступивших в ВУЗы, если это допустимо

- найдено количество медалистов

- запись информации в файл.

1. Тексты модуля, программы
   1. Модуль School

Файл School.h

#pragma once

#include <vector>

#include <fstream>

#include <iostream>

#include <string>

using namespace std;

enum SchoolKind

{

school=0, lyceum, gymnasium

};

class Info

{

public:

int Year; // Год

int CountAll; // Количество выпускников

int CountGoldMedal; // Количество золотых медалей

int CountSilverMedal; // Количество серебряных медалей

int CountHigher; // Количество поступивших в высшие заведения

int CountMiddle; // Количество поступивших в средние заведения

};

class School

{

public:

School();

~School();

string code; // Код заведения

SchoolKind kind; // Вид заведения

vector<Info> YearInfo; // Информация о выпусках по годам

void GetData(string str); // Получение данных из строки

void ToCSV(vector<string> &v); // Формирование массива строк по массиву данных

int countGoldMedals(int year); // Количество золотых медалистов, параметр - год

int countAll(int year); // Количество выпускников, параметр - год

int countHigher(int year); // Количество выпускников, поступивших в ВУЗы, параметр - год

};

Файл School.cpp

#include "stdafx.h"

#include "School.h"

using namespace std;

School::School()

{

}

School::~School()

{

}

int School::countGoldMedals(int year)

{

int num = YearInfo.size();

int sum = 0;

// Перебор по годам

for (int i = 0; i < num; ++i)

{

if(YearInfo[i].Year == year)

sum += YearInfo[i].CountGoldMedal; // Количество золотых медалей

}

return sum;

}

int School::countAll(int year)

{

int num = YearInfo.size();

int sum = 0;

// Перебор по годам

for (int i = 0; i < num; ++i)

{

if (YearInfo[i].Year == year)

sum += YearInfo[i].CountAll; // Количество выпускников

}

return sum;

}

int School::countHigher(int year)

{

int num = YearInfo.size();

int sum = 0;

// Перебор по годам

for (int i = 0; i < num; ++i)

{

if (YearInfo[i].Year == year)

sum += YearInfo[i].CountHigher; // Количество выпускников поступивших в ВУЗы

}

return sum;

}

void School::GetData(string str)

{

//int nbuf = 256;

//char s1[256], s2[256], s3[256], s4[256], s5[256], s6[256];

vector<string> vec;

std::string::size\_type found0 = -1;

std::string::size\_type found1 = str.find\_first\_of(";");

while (found1 != std::string::npos)

{

string ss = str.substr(found0 + 1, found1 - found0 - 1);

vec.push\_back(ss);

found0 = found1;

found1 = str.find\_first\_of(";", found0 + 1);

}

code = vec[0];

Info info;

info.Year = stoi(vec[1]);

info.CountGoldMedal = stoi(vec[2]);

info.CountSilverMedal = stoi(vec[3]);

info.CountAll = stoi(vec[4]);

info.CountHigher = stoi(vec[5]);

info.CountMiddle = stoi(vec[6]);

YearInfo.push\_back(info);

}

void School::ToCSV(vector<string> &v)

{

char \*str = new char[256];

int num = YearInfo.size();

for (int i = 0; i < num; ++i)

{

string s = code + ";";

sprintf\_s(str, 256, "%d", YearInfo[i].Year);

s = s + str + ";";

sprintf\_s(str, 256, "%d", YearInfo[i].CountAll);

s = s + str + ";";

sprintf\_s(str, 256, "%d", YearInfo[i].CountGoldMedal);

s = s + str + ";";

sprintf\_s(str, 256, "%d", YearInfo[i].CountSilverMedal);

s = s + str + ";";

sprintf\_s(str, 256, "%d", YearInfo[i].CountHigher);

s = s + str + ";";

sprintf\_s(str, 256, "%d", YearInfo[i].CountMiddle);

s = s + str + ";";

v.push\_back(s);

}

return ;

}

* 1. Модуль Schools

Файл Schools.h

#pragma once

#include "School.h"

class Schools

{

public:

Schools();

~Schools();

bool ReadFromFile(const string FName);

bool WriteToFile(const string FName);

int countGoldMedals(int year);

int countAll(string code, int year);

int countHigher(string code, int year);

private:

vector<School> schools;

};

Файл Schools.cpp

#include "stdafx.h"

#include "Schools.h"

Schools::Schools()

{

}

Schools::~Schools()

{

}

int Schools::countAll(string code, int year)

{

int sum = 0;

int num = schools.size();

bool find = false;

int index = 0;

// Перебор всех учебных заведений

for (int i = 0; i < num; ++i)

{

find = schools[i].code == code;

if (find)

{

index = i; // Найдена

break;

}

}

if (find)

{

// Найдена - считаем

sum = schools[index].countAll(year);

}

return sum;

}

int Schools::countHigher(string code, int year)

{

int sum = 0;

int num = schools.size();

bool find = false;

int index = 0;

// Перебор всех учебных заведений

for (int i = 0; i < num; ++i)

{

find = schools[i].code == code;

if (find)

{

index = i; // Найдена

break;

}

}

if (find)

{

// Найдена - считаем

sum = schools[index].countHigher(year);

}

return sum;

}

int Schools::countGoldMedals(int year)

{

int num = schools.size();

int sum = 0;

// Перебор всех учебных заведений

for (int i = 0; i < num; ++i)

{

sum += schools[i].countGoldMedals(year); // Из каждого добываем некоторое количество строк

}

return sum;

}

bool Schools::ReadFromFile(const string FName)

{

ifstream FRead(FName);

bool ret = true;

int bufsize = 256;

char \*buff = new char[bufsize]; // буфер промежуточного хранения считываемого из файла текста

try

{

if (!FRead.is\_open()) // если файл не открыт

{

return false;

}

else

{

while (FRead.getline(buff, bufsize)) // Читаем строку из файла

{

string str = buff; // Перевод в string

std::string::size\_type found0 = -1;

std::string::size\_type found1 = str.find\_first\_of(";");

//К какой школе относится строка

string code = str.substr(found0 + 1, found1 - found0 - 1);

//Поиск школы с таким кодом. Если нет - добавить

int num = schools.size();

bool find = false;

School school;

int index = 0;

for (int i = 0; i < num; ++i)

{

find = schools[i].code == code;

if (find)

{

index = i; // Найдена

break;

}

}

if (!find)

{

// Не нацдена - добавляем

school.code = code;

schools.push\_back(school);

index = schools.size()-1; // Номер - последний

}

schools[index].GetData(str); // Запись данных

}

FRead.close(); // закрываем файл

}

}

catch (...)

{

ret = false;

}

delete buff;

}

bool Schools::WriteToFile(const string FName)

{

ofstream FWrite(FName);

bool ret = true;

int bufsize = 256;

char \*buff = new char[bufsize]; // буфер промежуточного хранения считываемого из файла текста

School school;

//int num = 0;

try

{

if (!FWrite.is\_open()) // если файл не открыт

{

return false;

}

else

{

int num = schools.size();

vector<string> text;

// Перебор всех учебных заведений

for (int i = 0; i < num; ++i)

{

schools[i].ToCSV(text); // Из каждого добываем некоторое количество строк

}

num = text.size();

// Все полученные строки по всем школам

for (int i = 0; i < num; ++i)

{

FWrite << text[i] << endl;

}

FWrite.close(); // закрываем файл

}

}

catch (...)

{

ret = false;

}

return ret;

}

* 1. Модуль SchoolsTest.cpp

// SchoolsTest.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <conio.h>

#include "School.h"

#include "Schools.h"

int main()

{

setlocale(LC\_ALL, "rus");

Schools sts;

sts.ReadFromFile("Schools.txt");

cout << "Введите год ";

int year;

cin >> year;

int col = sts.countGoldMedals(year);

cout << "Количество золотых медалистов в " << year << " году " << col << endl;

string code = "001s01";

int numAll = sts.countAll(code, year);

cout << "Количество выпускников в " << year << " году в школе " << code << " " << numAll << endl;

int numH = sts.countHigher(code, year);

cout << "Количество выпускников в " << year << " году в школе " << code << " поступивших в ВУЗы " << numH << endl;

if (numAll != 0)

{

double fraction = 1.0\*numH / numAll \* 100;

cout << "Процент - " << fraction << "%" << endl;

}

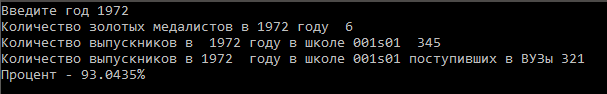
sts.WriteToFile("Schools.out");

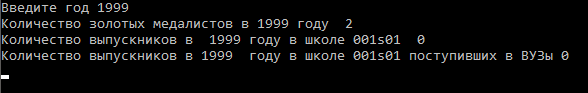
\_getch();

return 0;

}

1. Результат тестирования на различных данных





1. Список использованных источников
2. <http://www.cplusplus.com/>
3. <https://metanit.com/cpp/>
4. К. Джамса. Учимся программировать на языке C++: Пер. с англ. - Москва: Мир, 1997. 320с.