Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №7

СТАНДАРТНАЯ БИБЛИОТЕКА ШАБЛОНОВ.

КОНТЕЙНЕРЫ.

Выполнил: ст. гр. ИТб-21

Куркчи А.Э.

Проверила:

Тимофеев И.С.

Севастополь

2015

# 1. Цель

Приобретение практических навыков в написании объектно-ориентированных программ с использованием контейнеров стандартной библиотеки шаблонов. Освоение особенностей отладки объектно-ориентированных программ.

2. Постановка задач

Вариант №1

Написать программу, моделирующую управление каталогом в файловой системе. Для каждого файла в каталоге содержатся следующие сведения: имя файла, дата создания, количество обращений к файлу.

Программа должна обеспечивать:

* Начальное формирование каталога файлов;
* Вывод каталога файлов;
* Удаление файлов, дата создания которых раньше заданной;
* Выборку файла с наибольшим количеством обращений.

Выбор моделируемой функции должен осуществляться с помощью меню. Для представления каталога использовать контейнерный класс list из STL.

3. Текст программы

#include <iostream>

#include <sstream>

#include <fstream>

#include <iomanip>

#include <string.h>

#include <algorithm>

#include <list>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

**using** **namespace** std**;**

struct datetime **{**

int year**;**

int month**;**

int day**;**

int hour**;**

int minute**;**

int second**;**

**};**

struct node\_s **{**

char name**[**255**];**

datetime created\_at**;**

unsigned long long hits**;**

**};**

class node **{**

private**:**

char name**[**255**];**

datetime created\_at**;**

unsigned long long hits**;**

public**:**

node**()** **{**

cout **<<** "Name:" **<<** endl**;**

cin **>>** name**;**

cout **<<** "Date & time (dd mm yyyy hh mm ss):" **<<** endl**;**

cin **>>** created\_at**.**day **>>** created\_at**.**month **>>** created\_at**.**year **>>** created\_at**.**hour **>>** created\_at**.**minute **>>** created\_at**.**second**;**

cout **<<** "Views:" **<<** endl**;**

cin **>>** hits**;**

**}**

node**(**const node\_s**&** s**)** **{**

strcpy**(**name**,**s**.**name**);**

created\_at **=** s**.**created\_at**;**

hits **=** s**.**hits**;**

**}**

node**(**const char**\*** n**,**const datetime c**,**const unsigned long long h **=** 0**)** **{**

strcpy**(**name**,**n**);**

created\_at **=** c**;**

hits **=** h**;**

**}**

const char**\*** get\_name**()** const **{**

**return** name**;**

**}**

datetime get\_created\_at**()** const **{**

**return** created\_at**;**

**}**

unsigned long long get\_hits**()** const **{**

**return** hits**;**

**}**

node\_s to\_struct**()** **{**

node\_s t**;**

strcpy**(**t**.**name**,**name**);**

t**.**created\_at **=** created\_at**;**

t**.**hits **=** hits**;**

**return** t**;**

**}**

void hit**()** **{**

hits**++;**

**}**

**};**

class directory **{**

private**:**

list**<**node**>** contents**;**

char fname**[**255**];**

const char**\*** format\_time**(**datetime t**)** **{**

stringstream ss**;**

ss **<<** setfill**(**'0'**)** **<<** setw**(**2**)** **<<** t**.**day **<<** "." **<<** setw**(**2**)** **<<** t**.**month **<<** "." **<<** setw**(**4**)** **<<** t**.**year **<<** " " **<<** setw**(**2**)** **<<** t**.**hour **<<** ":" **<<** setw**(**2**)** **<<** t**.**minute **<<** ":" **<<** setw**(**2**)** **<<** t**.**second**;**

**return** ss**.**str**().**c\_str**();**

**}**

int compare\_time**(**datetime a**,**datetime b**)** **{**

**if(**a**.**year **!=** b**.**year**)** **{**

**return** a**.**year **-** b**.**year**;**

**}**

**if(**a**.**month **!=** b**.**month**)** **{**

**return** a**.**month **-** b**.**month**;**

**}**

**if(**a**.**day **!=** b**.**day**)** **{**

**return** a**.**day **-** b**.**day**;**

**}**

**if(**a**.**hour **!=** b**.**hour**)** **{**

**return** a**.**hour **-** b**.**hour**;**

**}**

**if(**a**.**minute **!=** b**.**minute**)** **{**

**return** a**.**minute **-** b**.**minute**;**

**}**

**if(**a**.**second **!=** b**.**second**)** **{**

**return** a**.**second **-** b**.**second**;**

**}**

**return** 0**;**

**}**

public**:**

directory**(**const char**\*** f**)** **{**

strcpy**(**fname**,**f**);**

**}**

void print**()** **{**

**if(**contents**.**empty**())** **{**

cout **<<** "Directory is empty" **<<** endl**;**

**return;**

**}**

list**<**node**>::**iterator it**;**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

it**->**hit**();**

cout **<<** format\_time**(**it**->**get\_created\_at**())** **<<** ": " **<<** it**->**get\_name**()** **<<** "; Views: " **<<** it**->**get\_hits**()** **<<** endl**;**

**}**

**}**

void print\_views**()** **{**

**if(**contents**.**empty**())** **{**

cout **<<** "Directory is empty" **<<** endl**;**

**return;**

**}**

list**<**node**>::**iterator it**;**

unsigned long long views **=** 0**;**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

**if(**it**->**get\_hits**()** **>** views**)** **{**

views **=** it**->**get\_hits**();**

**}**

**}**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

**if(**it**->**get\_hits**()** **==** views**)** **{**

views **=** it**->**get\_hits**();**

cout **<<** format\_time**(**it**->**get\_created\_at**())** **<<** ": " **<<** it**->**get\_name**()** **<<** "; Views: " **<<** it**->**get\_hits**()** **<<** endl**;**

**}**

**}**

**}**

void add**(**const node**&** f**)** **{**

list**<**node**>::**iterator it**;**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

**if(**strcmp**(**it**->**get\_name**(),**f**.**get\_name**())** **==** 0**)** **{**

contents**.**erase**(**it**);**

**break;**

**}**

**}**

contents**.**push\_back**(**f**);**

**}**

void remove**(**const char**\*** n**)** **{**

**if(**contents**.**empty**())** **{**

cout **<<** "Directory is empty" **<<** endl**;**

**return;**

**}**

list**<**node**>::**iterator it**;**

bool found **=** **false;**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

**if(**strcmp**(**it**->**get\_name**(),**n**)** **==** 0**)** **{**

contents**.**erase**(**it**);**

found **=** **true;**

**break;**

**}**

**}**

**if(!**found**)** **{**

cout **<<** "No such file" **<<** endl**;**

**}**

**}**

void remove**(**datetime t**)** **{**

**if(**contents**.**empty**())** **{**

cout **<<** "Directory is empty" **<<** endl**;**

**return;**

**}**

list**<**node**>::**iterator it**;**

bool found **=** **false;**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

**if(**compare\_time**(**it**->**get\_created\_at**(),**t**)** **<=** 0**)** **{**

contents**.**erase**(**it**);**

found **=** **true;**

**break;**

**}**

**}**

**if(!**found**)** **{**

cout **<<** "No such file" **<<** endl**;**

**}**

**}**

void read**()** **{**

contents**.**erase**(**contents**.**begin**(),**contents**.**end**());**

ifstream fin**(**fname**,**ios**::**binary**);**

node\_s st**;**

**while(!**fin**.**eof**())** **{**

fin**.**read**((**char**\*)** **&**st**,sizeof(**node\_s**));**

add**(**node**(**st**));**

**}**

fin**.**close**();**

**}**

void save**()** **{**

ofstream fout**(**fname**,**ios**::**binary**);**

list**<**node**>::**iterator it**;**

**for(**it **=** contents**.**begin**();**it**!=**contents**.**end**();++**it**)** **{**

node\_s st **=** it**->**to\_struct**();**

fout**.**write**((**char**\*)&**st**,sizeof(**node\_s**));**

**}**

fout**.**close**();**

**}**

**};**

int main**()** **{**

cout **<<** "SevGU semestr.03 OOP lab.07" **<<** endl**;**

char fname**[**255**];**

cout **<<** "Savefile name:" **<<** endl**;**

cin **>>** fname**;**

directory dir**(**fname**);**

char name**[**255**];**

datetime from**;**

char key**;**

**do** **{**

system**(**"clear"**);**

cout **<<** "SevGU semestr.03 OOP lab.07" **<<** endl**;**

cout **<<** "Menu" **<<** endl**;**

cout **<<** "\t1.View directory contents" **<<** endl**;**

cout **<<** "\t2.View most viewed" **<<** endl**;**

cout **<<** "\t3.Add file to directory" **<<** endl**;**

cout **<<** "\t4.Delete file from directory (by name)" **<<** endl**;;**

cout **<<** "\t5.Delete all files created before" **<<** endl**;**

cout **<<** "\t6.Read from file" **<<** endl**;**

cout **<<** "\t7.Save to file" **<<** endl**;**

cout **<<** endl**;**

cout **<<** "\t0.Quit" **<<** endl**;**

cin **>>** key**;**

system**(**"clear"**);**

**switch(**key**)** **{**

**case** '1'**:**

dir**.**print**();**

cout **<<** "Press Enter to continue..." **<<** endl**;**

cin**.**get**();**

cin**.**get**();**

**break;**

**case** '2'**:**

dir**.**print\_views**();**

cout **<<** "Press Enter to continue..." **<<** endl**;**

cin**.**get**();**

cin**.**get**();**

**break;**

**case** '3'**:**

dir**.**add**(**node**());**

**break;**

**case** '4'**:**

cout **<<** "Name:" **<<** endl**;**

cin **>>** name**;**

dir**.**remove**(**name**);**

**break;**

**case** '5'**:**

cout **<<** "Date & time (dd mm yyyy hh mm ss):" **<<** endl**;**

cin **>>** from**.**day **>>** from**.**month **>>** from**.**year **>>** from**.**hour **>>** from**.**minute **>>** from**.**second**;**

dir**.**remove**(**from**);**

**break;**

**case** '6'**:**

dir**.**read**();**

**break;**

**case** '7'**:**

dir**.**save**();**

**break;**

**}**

**}** **while(**key **!=** '0'**);**

**return** 0**;**

**}**

4. Тестовые примеры

На рисунке 1 представлено меню программы

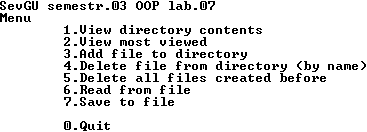


Рисунок 1 – Меню программы

На рисунке 2 представлен вывод первого пункта меню (просмотр всех файлов)



Рисунок 2 – Просмотр всех файлов

На рисунке 3 представлен вывод второго пункта меню (просмотр самых просматриваемых файлов)



Рисунок 3 – Просмотр самых просматриваемых файлов

Вывод

В ходе лабораторной работы были приобретены практические навыки в написании объектно-ориентированных программ с использованием контейнеров стандартной библиотеки шаблонов. Освоены особенности отладки объектно-ориентированных программ.

Приобретение практических навыков в написании объектно-ориентированных программ с использованием контейнеров стандартной библиотеки шаблонов. Освоение особенностей отладки объектно-ориентированных программ.