Министерство образования и науки Российской Федерации

ФГАОУ ВПО «УрФУ имени первого Президента России Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РтФ

Кафедра радиоэлектроники и информационных систем

**Отчет по лабораторной работе №6**

Преподаватель Бабич Михаил Владимирович

Студенты гр. РИ-240002 Овчинникова К. С. Трефилов А.А.

Екатеринбург

2015

Задача 5

Задача: Прочитайте файл safeArray.cpp. Сделайте из класса safearray шаблон, так, чтобы он мог хранить любые типы данных. В секции main создайте по крайней мере два массива различных типов.

Код:

#include <iostream>

using namespace std;

#include <process.h>

const int LIMIT = 5;

template <class T>

class safearay

{

private:

T arr[LIMIT];

public:

T& operator [](int n)

{

if (n< 0 || n >= LIMIT)

{

cout << "\nIndex out of bounds"; exit(1);

}

return arr[n];

}

};

////////////////////////////////////////////////////////////////

void main()

{

char choice;

cout << "Type? (int - i; char - c): ";

cin >> choice;

if (choice == 'c')

{

safearay <char> sa1;

for (int j = 0; j<LIMIT; j++)

{

cout << j << ": ";

cin >> sa1[j];

}

for (int j = 0; j<LIMIT; j++)

{

char temp = sa1[j];

cout << "Element " << j << " is " << temp << endl;

}

}

if (choice == 'i')

{

safearay <int> sa1;

for (int j = 0; j<LIMIT; j++)

{

cout << j << ": ";

cin >> sa1[j];

}

for (int j = 0; j<LIMIT; j++)

{

char temp = sa1[j];

cout << "Element " << j << " is " << temp << endl;

}

}

system("pause");

}

Задача 10

Задача: В цикле запрашивайте у пользователя данные, состоящие из имени, отчества, фамилии и номера работника (типа unsigned long). Затем осуществите форматированный вывод в объект ofstream с помощью оператора вставки («). Не забывайте, что строки данных должны оканчиваться пробелами или другими разделителями. Когда пользователь сообщит об окончании ввода, закройте объект ofstream, откройте объект ifstream, прочитайте и выведите на экран все данные из файла, после чего завершите программу.

Код:

#include <iostream>

#include <clocale>

#include <string>

#include <vector>

#include <fstream>

#include <iomanip>

unsigned short int workers = 0;

void input();

void input\_file();

void output\_file();

unsigned long number[100];

vector <string> first\_name[100], middle\_name[100], family\_name[100];

using namespace std;

void main()

{

setlocale(LC\_ALL, "Russian");

input();

output\_file();

system("pause");

}

void input()

{

char choice;

cout << "Имя: ";

cin >> first\_name[workers];

cout << "Отчество: ";

cin >> middle\_name[workers];

cout << "Фамилия: ";

cin >> family\_name[workers];

cout << "Номер: ";

cin >> number[workers];

cout << "Продолжить ввод? (y/n): ";

cin >> choice;

workers++;

if (choice == 'y')

input();

}

void input\_file()

{

ofstream fin("file.txt");

fin << setw(12) << setiosflags(ios::left) << " " << setw(12)

<< setiosflags(ios::left) << "Имя: " << setw(12)

<< setiosflags(ios::left) << "Отчество: " << setw(12)

<< setiosflags(ios::left) << "Фамилия: " << setw(12)

<< setiosflags(ios::left) << "Номер: " << setw(12) << endl;

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

{

fin << setw(12) << setiosflags(ios::left) << i << setw(12)

<< setiosflags(ios::left) << first\_name[i] << setw(12)

<< setiosflags(ios::left) << middle\_name[i] << setw(12)

<< setiosflags(ios::left) << family\_name[i] << setw(12)

<< setiosflags(ios::left) << number[i] << setw(12) << endl;

}

fin.close();

}

void output\_file()

{

char buff[80];

ifstream fin("file.txt");

fin >> buff;

cout << buff << endl;

fin.getline(buff, 80);

fin.close();

cout << buff << endl;

Задача 1

Задание: Прочитайте файл linkedList.cpp. Допустим, мы заполняем объект этого класса данными, а затем присваиваем один класс целиком другому, используя стандартный оператор присваивания:  
list2 = list1;  
Допустим, что впоследствии мы удалим объект класса list1. Можем ли мы все еще использовать list2 для доступа к введенным данным? Увы, нет, так как при удалении list1 все его ссылки были удалены. Единственное, что было известно объекту linklist про удаленный объект, это указатель на него. Но его удалили, указатель в list2 стал недееспособным, и все попытки получить доступ к данным приведут к получению мусора вместо данных, а в худшем случае — к зависанию программы. Один из способов избежать этих проблем — перегрузить оператор присваивания, чтобы он вместе с объектом копировал бы все его ссылки. Но тогда придется пройти по всей цепочке, поочередно копируя все ссылки. Следует также перегружать конструктор копирования. Чтобы была возможность удалять объекты linklist в main(), можно создавать их с помощью указателя и new. В таком случае проще будет проверять работу новых операций.

Код:

#include "stdafx.h"

#include <iostream>

#include <Windows.h>

using namespace std;

struct link

{

int data;

link\* next;

};

class linklist

{

private:

link\* first;

public:

linklist()

{ first = NULL; }

linklist(const linklist& from)

{

first = NULL;

link\* current = from.first;

while( current != NULL )

{

link\* newlink = new link;

newlink->data = current->data;

newlink->next = first;

first = newlink;

current = current->next;

}

}

~linklist();

linklist & operator=(const linklist & from);

void additem(int d);

void display();

};

void linklist::additem(int d)

{

link\* newlink = new link;

newlink->data = d;

newlink->next = first;

first = newlink;

}

void linklist::display()

{

link\* current = first;

while( current != NULL )

{

cout << current->data << endl;

current = current->next;

}

}

linklist::~linklist()

{

link\* current = first;

while( current != NULL )

{

link\* temp = current;

current = current->next;

delete temp;

}

}

linklist & linklist::operator=(const linklist & from)

{

link\* current = from.first;

while( current != NULL )

{

link\* newlink = new link;

newlink->data = current->data;

newlink->next = first;

first = newlink;

current = current->next;

}

return \*this;

}

int main()

{

linklist\* li = new linklist;

linklist\* li2= new linklist;

li->additem(25);

li->additem(36);

li->additem(49);

li->additem(64);

cout << "Вывод элементов первого списка: " << endl;

li->display();

\*li2=\*li;

delete li;

cout << "Копирование элементов первого списка во второй. Удаление первого списка. Вывод элементов второго списка: " << endl;

li2->display();

system("PAUSE");

return 0;

}

Задача 6

Задание: Класс дроби из упражнения 2 преобразуйте в шаблон, так, чтобы дробь могла хранить числитель и знаменатель различных типов.

Код:

#include <iostream>

#include <Windows.h>

using namespace std;

template <class T>

class frac

{

private:

T num;

T denom;

public:

frac(){};

frac(T num1)

{

num = num1;

denom = 1;

}

frac(T num1, T denom1)

{

num = num1;

denom = denom1;

}

void output() const

{

cout << " = " << num << "/" << denom << endl << endl;

}

friend frac<T> operator+(const frac<T> & frac1, const frac<T> & frac2)

{

frac<T> frac3;

frac3.num = frac1.num \* frac2.denom + frac2.num \* frac1.denom;

frac3.denom = frac1.denom \* frac2.denom;

return frac3;

}

friend frac<T> operator-(const frac<T> & frac1, const frac<T> & frac2)

{

frac<T> frac3;

frac3.num = frac1.num \* frac2.denom - frac2.num \* frac1.denom;

frac3.denom = frac1.denom \* frac2.denom;

return frac3;

}

friend frac<T> operator\*(const frac<T> & frac1, const frac<T> & frac2)

{

frac<T> frac3;

frac3.num = frac1.num \* frac2.num;

frac3.denom = frac1.denom \* frac2.denom;

return frac3;

}

friend frac<T> operator/(const frac<T> & frac1, const frac<T> & frac2)

{

frac<T> frac3;

frac3.num = frac1.num \* frac2.denom;

frac3.denom = frac1.denom \* frac2.num;

return frac3;

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

frac<int> frac1(7, 5), frac2(8, 4);

frac<int> frac3 = frac1 + 5;

frac3.output();

frac<int> frac4 = 5 + frac2;

frac4.output();

frac<int> frac5 = frac1 + frac2;

frac5.output();

///////////////////////////////////////////////////////

frac<double> frac6(7, 5), frac7(8, 4);

frac<double> frac8 = frac6 + 5.7;

frac8.output();

frac<double> frac9 = 5.2 + frac7;

frac9.output();

frac<double> frac10 = frac6 + frac7;

frac10.output();

system("pause");

return 0;

}

**Вывод**

В данной лабораторной работе мы научились работать с шаблонами, перегрузками операторов и файлами.