ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

*Институт Принтмедиа и информационных технологий*

*Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»,

ЛАБОРАТОРНАЯ РАБОТА № 8

Дисциплина: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Тема: Наследование

Выполнил(а): студент(ка) группы 181-722

Голиков Р.О.

(Фамилия И.О.)

Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

Проверил: Ктн Доцент Арсентьев Д.А.*\_\_\_\_\_\_\_\_\_\_*

(Фамилия И.О. степень, звание) (Оценка)

Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

2019

**Задание 1**

Представьте себе издательскую компанию, которая торгует книгами и аудиозаписями этих книг. Создайте класс publication, в котором хранятся название (строка) и цена (типа float) книги. От этого класса наследуются еще два класса: book, который содержит информацию о количестве страниц в книге (типа int), и type, который содержит время записи книги в минутах (тип float). В каждом из этих трех классов должен быть метод getdata(), через который можно получать данные от пользователя с клавиатуры, и putdata(), предназначенный для вывода этих данных.

Напишите функцию main() программы для проверки классов book и type. Создайте их объекты в программе и запросите пользователя ввести и вывести данные с использованием методов getdataQ и putdata().

**Код:**

#include <iostream>

#include <string>

using namespace std;

class publication // base class

{

private:

string title;

float price;

public:

void getdata()

{

cout << "\nEnter title: "; cin >> title;

cout << "Enter price: "; cin >> price;

}

void putdata() const

{

cout << "\nTitle: " << title;

cout << "\nPrice: " << price;

}

};

class book : private publication // derived class

{

private:

int pages;

public:

void getdata()

{

publication::getdata();

cout << "Enter number of pages: "; cin >> pages;

}

void putdata() const

{

publication::putdata();

cout << "\nPages: " << pages;

}

};

class tape : private publication // derived class

{

private:

float time;

public:

void getdata()

{

publication::getdata();

cout << "Enter playing time: "; cin >> time;

}

void putdata() const

{

publication::putdata();

cout << "\nPlaying time: " << time;

}

};

int main()

{

book book1; // define publications

tape tape1;

book1.getdata(); // get data for them

tape1.getdata();

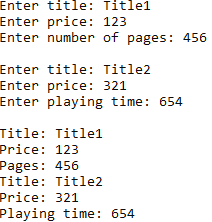
book1.putdata(); // display their data

tape1.putdata();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 2**

Пусть SZ – константа, которая обозначает количество символов в строке. Создайте класс Pstring, производный от класса String, в котором предотвратите возможность переполнения буфера при определении слишком длинной строковой константы. Класс String, соответственно, должен иметь в поле, котором хранится строка, ограничение на количество символов в строке (буфер ограниченной длины). Новый конструктор производного класса будет копировать в str только SZ-1 символов, если строка окажется слишком длинной, и будет копировать строку полностью, если она будет иметь длину меньшую, чем SZ. Напишите функцию main() программы для проверки ее работы со строками разной длины.

**Код:**

#include <iostream>

#include <cstring> //for strcpy(), etc.

using namespace std;

class String //base class

{

protected: //Note: can't be private

enum { SZ = 80 }; //size of all String objects

char str[SZ]; //holds a C-string

public:

String() //constructor 0, no args

{

str[0] = '\0';

}

String(const char s[]) //constructor 1, one arg

{

strcpy\_s(str, s);

} // convert string to String

void display() const //display the String

{

cout << str;

}

operator char\*() //conversion function

{

return str;

} //convert String to C-string

};

class Pstring : public String //derived class

{

public:

Pstring(const char s[]) //constructor for Pstring

{

if (strlen(s) > SZ - 1) //if too long,

{

int j;

for (j = 0; j < SZ - 1; j++) //copy the first SZ-1

str[j] = s[j]; //characters "by hand"

str[j] = '\0'; //add the null character

}

else //not too long,

strcpy\_s(str, s);

}

};

int main()

{ //define String

Pstring s1 = "This is a very long string which is probably no, certainly--going to exceed the limit set by SZ.";

cout << "\ns1="; s1.display(); //display String

Pstring s2 = "This is a short string.";

cout << "\ns2="; s2.display(); //display String

cout << endl;

return 0;

}

**Вывод:**  


**Задание 3**

Начните с классов book, type и publication из упражнения 1. Добавьте базовый класс sales, в котором содержится массив, состоящий из трех значений типа float, куда можно записать общую стоимость проданных книг за последние три месяца. Включите в класс методы getdata() для получения значений стоимости от пользователя и putdata() для вывода этих цифр. Измените классы book и type так, чтобы они стали производными обоих классов: publications и sales. Объекты классов book и type должны вводить и выводить данные о продажах вместе с другими своими данными. Напишите функцию main() для создания объектов классов book и type, чтобы протестировать возможности ввода/вывода данных.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class publication

{

private:

string title;

float price;

public:

void getdata()

{

cout << "\nEnter title: "; cin >> title;

cout << " Enter price: "; cin >> price;

}

void putdata() const

{

cout << "\nTitle: " << title;

cout << "\n Price: " << price;

}

};

class sales

{

private:

enum { MONTHS = 3 };

float salesArr[MONTHS];

public:

void getdata();

void putdata() const;

};

void sales::getdata()

{

cout << " Enter sales for 3 months\n";

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

{

cout << " Month " << j + 1 << ": ";

cin >> salesArr[j];

}

}

void sales::putdata() const

{

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

{

cout << "\n Sales for month " << j + 1 << ": ";

cout << salesArr[j];

}

}

class book : private publication, private sales

{

private:

int pages;

public:

void getdata()

{

publication::getdata();

cout << " Enter number of pages: "; cin >> pages;

sales::getdata();

}

void putdata() const

{

publication::putdata();

cout << "\n Pages: " << pages;

sales::putdata();

}

};

class tape : private publication, private sales

{

private:

float time;

public:

void getdata()

{

publication::getdata();

cout << " Enter playing time: "; cin >> time;

sales::getdata();

}

void putdata() const

{

publication::putdata();

cout << "\n Playing time: " << time;

sales::putdata();

}

};

int main()

{

book book1; // define publications

tape tape1;

book1.getdata(); // get data for publications

tape1.getdata();

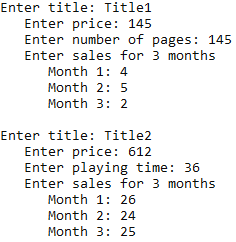
book1.putdata(); // display data for publications

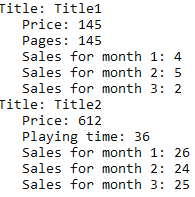
tape1.putdata();

cout << endl;

return 0;

}

**Вывод:**  




**Задание 4**

Предположим, что издатель из упражнений 1 и 3 решил добавить к своей продукции версии книг на компьютерных дисках для тех, кто любит читать книги на своих компьютерах. Добавьте класс disk, который, как book и type, является производным класса publication. Класс disk должен включать в себя те же функции, что и в других классах. Полем только этого класса будет тип диска: CD или DVD. Для хранения этих данных вы можете ввести тип enum. Пользователь должен выбрать подходящий тип, набрав на клавиатуре с или d.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class publication

{

private:

string title;

float price;

public:

void getdata()

{

cout << "\nEnter title: "; cin >> title;

cout << " Enter price: "; cin >> price;

}

void putdata() const

{

cout << "\nTitle: " << title;

cout << "\n Price: " << price;

}

};

class sales

{

private:

enum { MONTHS = 3 };

float salesArr[MONTHS];

public:

void getdata();

void putdata() const;

};

void sales::getdata()

{

cout << " Enter sales for 3 months\n";

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

{

cout << " Month " << j + 1 << ": ";

cin >> salesArr[j];

}

}

void sales::putdata() const

{

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

{

cout << "\n Sales for month " << j + 1 << ": ";

cout << salesArr[j];

}

}

class disk : private publication, private sales

{

private:

enum type {DVD, CD } \_type;

public:

void getdata()

{

publication::getdata();

char ch;

cout << " Enter type of disk(c/d): ";

cin >> ch;

switch (ch)

{

case 'c': \_type = CD; break;

case 'd': \_type = DVD; break;

default: cout << "Неизвестный тип.\n"; break;

}

sales::getdata();

}

void putdata() const

{

publication::putdata();

cout << "\n Disk type: ";

switch (\_type)

{

case CD: cout << "CD"; break;

case DVD: cout << "DVD"; break;

default: cout << "Неизвестный тип.\n"; break;

}

sales::putdata();

}

};

int main()

{

disk d;

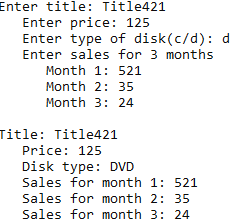
d.getdata();

d.putdata();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 5**

Создайте производный класс employee2 от базового класса employee (приведён ниже). Добавьте в новый класс поле compensation типа double и поле period типа enum для обозначения периода оплаты работы служащего: почасовая, понедельная или помесячная.

**Код:**

#include <iostream>

using namespace std;

const int LEN = 80;

class employee

{

private:

char name[LEN];

unsigned long number;

public:

void getdata()

{

cout << "\n Enter last name: "; cin >> name;

cout << " Enter number: "; cin >> number;

}

void putdata() const

{

cout << "\n Name: " << name;

cout << "\n Number: " << number;

}

};

class employee2 : private employee

{

private:

double compensation;

enum payperiod { Hourly, Weekly, Monthly } period;

public:

void getdata()

{

employee::getdata();

cout << "\n Enter compensation: "; cin >> compensation;

char ch;

cout << " Enter pay period(h/w/m): "; cin >> ch;

switch (ch)

{

case 'h': period = Hourly; break;

case 'w': period = Weekly; break;

case 'm': period = Monthly; break;

default: cout << "Неизвестный тип.\n"; break;

}

}

void putdata() const

{

employee::putdata();

cout << "\n Compensation: "; cout << compensation;

cout << "\n Disk type: ";

switch (period)

{

case Hourly: cout << "Hourly"; break;

case Weekly: cout << "Weekly"; break;

case Monthly: cout << "Monthly"; break;

default: cout << "Неизвестный тип.\n"; break;

}

}

};

int main()

{

employee2 e;

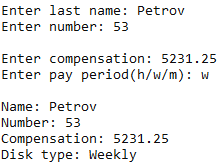
e.getdata();

e.putdata();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 6**

Используя наследование, добавьте к классу, который приведён ниже, возможность для пользователя определять верхнюю и нижнюю границы массива в конструкторе.

**Код:**

#include <iostream>

using namespace std;

const int LIMIT = 100;

class safearay

{

private:

int arr[LIMIT];

public:

int& operator[](int n)

{

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

{

cout << "Index out of bounds"; exit(1);

}

return arr[n];

}

};

class custarray : public safearay

{

private:

int lborder;

int uborder;

public:

custarray(int lowborder, int upperborder) : lborder(lowborder), uborder(upperborder)

{

}

int& operator[](int n)

{

return safearay::operator[](n >= uborder ? LIMIT : n - lborder);

}

};

int main()

{

custarray arr(100, 115);

for (int i = 100; i < 115; i++)

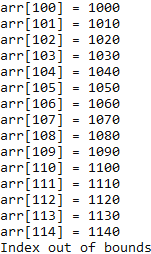
arr[i] = i \* 10;

for (int i = 100; i <= 115; i++)

cout << "arr[" << i << "] = " << arr[i] << endl;

return 0;

}

**Вывод:**  


**Задание 7**

Используя наследование на приведённом ниже классе, добавьте возможность использования постфиксных операций для случаев увеличения и уменьшения.

**Код:**

#include <iostream>

using namespace std;

class Counter

{

protected:

unsigned int count;

public:

Counter() : count(0)

{}

Counter(int c) : count(c)

{}

unsigned int get\_count() const

{

return count;

}

Counter operator++()

{

return Counter(++count);

}

};

class CountDn : public Counter

{

public:

CountDn() : Counter()

{ }

CountDn(int c) : Counter(c)

{ }

CountDn operator--()

{

return CountDn(--count);

}

};

class CountPost : public CountDn

{

public:

CountPost() : CountDn()

{ }

CountPost(int c) : CountDn(c)

{ }

CountPost operator++(int)

{

CountPost temp = \*this;

++count;

return temp;

}

CountPost operator--(int)

{

CountPost temp = \*this;

--count;

return temp;

}

};

int main()

{

CountPost c1;

CountPost c2(100);

cout << "\nc1 = " << c1.get\_count();

cout << "\nc2 = " << c2.get\_count();

cout << "\nc1++ = " << c1++.get\_count();

cout << "\nc2-- = " << c2--.get\_count();

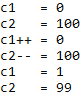
cout << "\nc1 = " << c1.get\_count();

cout << "\nc2 = " << c2.get\_count();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 8**

В некоторых компьютерных языках, таких, как Visual Basic, есть операции, с помощью которых можно выделить часть строки и присвоить ее другой строке. (В стандартном классе string предложены различные подходы). Используя наследование, добавьте такую возможность в класс Pstring из упражнения 2. В новом производном классе Pstring2 разместите три новых функции: left(), mid() и right(), которые будут принимать: Left() – один аргумент, количество символов, которое будет вырезано с левого края, Right() – один аргумент, количество символов, которое будет вырезано с правого края, Mid() – два аргумента. Первый – количество символов, которое будет вырезано, второй – с какого символа начинать резать.

**Код:**

#include <iostream>

#include <cstring>

using namespace std;

class String

{

protected:

enum { SZ = 80 };

char str[SZ];

public:

String()

{

str[0] = '\0';

}

String(const char s[])

{

strcpy\_s(str, s);

}

void display() const

{

cout << str;

}

operator char\*()

{

return str;

}

};

class Pstring : public String

{

public:

Pstring(const char s[])

{

if (strlen(s) > SZ - 1)

{

int j;

for (j = 0; j < SZ - 1; j++)

str[j] = s[j];

str[j] = '\0';

}

else

strcpy\_s(str, s);

}

};

class Pstring2 : public Pstring

{

public:

Pstring2(const char s[]) : Pstring(s)

{

}

Pstring2 left(int number)

{

Pstring2 substr = "";

int i;

for (i = 0; i < number; i++)

substr[i] = str[i];

substr[i] = '\0';

return substr;

}

Pstring2 right(int number)

{

Pstring2 substr = "";

int i;

int strend = SZ;

for (i = 0; i < SZ; i++)

if (str[i] == '\0') strend = i;

for (i = 0; i < number; i++)

substr[i] = str[i + strend - number];

substr[i] = '\0';

return substr;

}

Pstring2 mid(int position, int number)

{

Pstring2 substr = "";

int i;

for (i = 0; i < SZ - position && i < number; i++)

substr[i] = str[i + position];

substr[i] = '\0';

return substr;

}

};

int main()

{

Pstring2 s1 = "Hello world!";

cout << "\nleft = "; s1.left(5).display();

cout << "\nright = "; s1.right(6).display();

cout << "\nmid = "; s1.mid(3, 5).display();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 9**

Вспомним классы publication, book и type из упражнения 1. Предположим, что мы хотим добавить в классы book и type дату выхода книги. Создайте новый производный класс publication2, который является производным класса publication и включает в себя поле, хранящее эту дату. Затем измените классы book и type так, чтобы они стали производными класса publication2 вместо publication. Сделайте необходимые изменения функций классов так, чтобы пользователь мог вводить и выводить дату выхода книги.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class publication // base class

{

private:

string title;

float price;

public:

void getdata()

{

cout << "\nEnter title: "; cin >> title;

cout << "Enter price: "; cin >> price;

}

void putdata() const

{

cout << "\nTitle: " << title;

cout << "\nPrice: " << price;

}

};

class publication2 : public publication

{

private:

string date;

public:

void getdata()

{

publication::getdata();

cout << "Enter date: "; cin >> date;

}

void putdata() const

{

publication::putdata();

cout << "\nDate: " << date;

}

};

class book : private publication2 // derived class

{

private:

int pages;

public:

void getdata()

{

publication2::getdata();

cout << "Enter number of pages: "; cin >> pages;

}

void putdata() const

{

publication2::putdata();

cout << "\nPages: " << pages;

}

};

class tape : private publication2 // derived class

{

private:

float time;

public:

void getdata()

{

publication2::getdata();

cout << "Enter playing time: "; cin >> time;

}

void putdata() const

{

publication2::putdata();

cout << "\nPlaying time: " << time;

}

};

int main()

{

book book1; // define publications

tape tape1;

book1.getdata(); // get data for them

tape1.getdata();

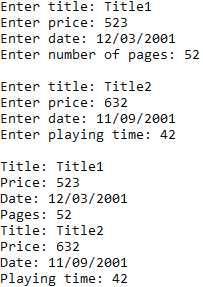
book1.putdata(); // display their data

tape1.putdata();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 10**

Создайте производный от класса manager класс executive. (Мы предполагаем, что управляющий — это главный менеджер.) Добавочными данными этого класса будут размер годовой премии и количество его акций в компании. Добавьте подходящие методы для этих данных, позволяющие их вводить и выводить.

**Код:**

#include <iostream>

using namespace std;

const int LEN = 80; //maximum length of names

class student //educational background

{

private:

char school[LEN]; //name of school or university

char degree[LEN]; //highest degree earned

public:

void getedu()

{

cout << " Enter name of school or university: ";

cin >> school;

cout << " Enter highest degree earned \n";

cout << " (Highschool, Bachelor's, Master's, PhD): ";

cin >> degree;

}

void putedu() const

{

cout << "\n School or university: " << school;

cout << "\n Highest degree earned: " << degree;

}

};

class employee

{

private:

char name[LEN]; //employee name

unsigned long number; //employee number

public:

void getdata()

{

cout << "\n Enter last name: "; cin >> name;

cout << " Enter number: "; cin >> number;

}

void putdata() const

{

cout << "\n Name: " << name;

cout << "\n Number: " << number;

}

};

class manager : private employee, private student //management

{

private:

char title[LEN]; //"vice-president" etc.

double dues; //golf club dues

public:

void getdata()

{

employee::getdata();

cout << " Enter title: "; cin >> title;

cout << " Enter golf club dues: "; cin >> dues;

student::getedu();

}

void putdata() const

{

employee::putdata();

cout << "\n Title: " << title;

cout << "\n Golf club dues: " << dues;

student::putedu();

}

};

class executive : public manager

{

private:

double annualpremium;

int stocks;

public:

void getdata()

{

manager::getdata();

cout << " Enter annual premium: "; cin >> annualpremium;

cout << " Enter stocks: "; cin >> stocks;

}

void putdata() const

{

manager::putdata();

cout << "\n Annual premium: " << annualpremium;

cout << "\n Stocks: " << stocks;

}

};

int main()

{

executive e;

cout << endl;

cout << "\nEnter data for executive";

e.getdata();

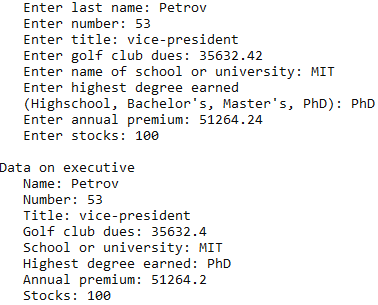
cout << "\nData on executive";

e.putdata();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 11**

В различных ситуациях иногда требуется работать с двумя числами, объединенными в блок. Например, каждая из координат экрана имеет горизонтальную составляющую (х) и вертикальную (у). Представьте такой блок чисел в качестве структуры pair, которая содержит две переменные типа int. Теперь предположим, что мы хотим иметь возможность хранить переменные типа pair в стеке. То есть мы хотим иметь возможность положить переменную тина pair в стек путем вызова метода push() с переменной типа pair в качестве аргумента и вынуть ее из стека путем вызова метода рор(), возвращающего переменную типа pair. Начнем с класса Stack2, что првиедён ниже. Создадим производный от него класс pairStack. В нем будут содержаться два метода: перегружаемый метод push() и перегружаемый метод рор(). Метод pairStack::push() должен будет сделать два вызова метода Stack2::push(), чтобы сохранить оба числа из пары, а метод pairStack::pop() должен будет сделать два вызова метода Stack2::pop().

**Код:**

#include <iostream>

using namespace std;

const int LIMIT = 100;

class Pair

{

public:

int Item1;

int Item2;

Pair() : Item1(0), Item2(0)

{}

Pair(int item1, int item2) : Item1(item1), Item2(item2)

{}

};

class Stack

{

private:

int \_arr[LIMIT];

int \_next;

public:

Stack() : \_next(0)

{}

void push(int el)

{

if (\_next == LIMIT)

{

cout << "\nStack overflow\n";

return;

}

\_arr[\_next++] = el;

}

int pop()

{

if (\_next == 0)

{

cout << "\nStack is empty\n";

return 0;

}

return \_arr[--\_next];

}

};

class pairStack : public Stack

{

public:

pairStack() : Stack()

{}

void push(Pair el)

{

Stack::push(el.Item1);

Stack::push(el.Item2);

}

Pair pop()

{

int item2 = Stack::pop();

int item1 = Stack::pop();

return Pair(item1, item2);

}

};

int main()

{

setlocale(LC\_ALL, "");

pairStack s;

Pair p;

for (int i = 0; i < 10; i++)

{

p.Item1 = i \* i + 3 \* i;

p.Item2 = i \* i - 2 \* i;

s.push(p);

cout << "В стэк положена пара: " << p.Item1 << " - " << p.Item2 << endl;

}

for (int i = 0; i < 10; i++)

{

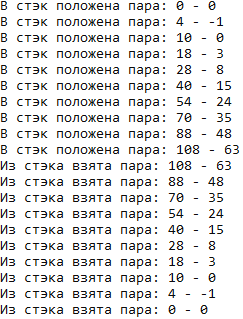
Pair newp = s.pop();

cout << "Из стэка взята пара: " << newp.Item1 << " - " << newp.Item2 << endl;

}

return 0;

}

**Вывод:**  


**Задание 12**

Рассмотрим старую Британскую платежную систему фунты-стерлинги-пенсы. Пенни в дальнейшем делятся на фартинги и полупенни. Фартинг — это 1/4 пенни. Существовали монеты фартинг, полфартиига и пенни. Любые сочетания монет выражались через восьмые части пенни.

Давайте предположим, что мы хотим добавить в класс sterling возможность пользоваться этими дробными частями пенни. Формат ввода/вывода может быть похожим на £1.1.1-1/4 или £9.19.11-7/8, где дефисы отделяют дробные части от пенни. Создайте новый класс sterfrac, производный от класса sterling. В нем должна быть возможность выполнения четырех основных арифметических операций со стерлингами, включая восьмые части пенни. Поле eighths типа int определяет количество восьмых. Вам нужно будет перегрузить методы класса sterling, чтобы они могли работать с восьмыми частями. Пользователь должен иметь возможность ввести и вывести дробные части пенни.

**Код:**

#include "Sterling.h"

#include <iostream>

using namespace std;

class sterfrac : public Sterling

{

private:

int \_eighths;

public:

sterfrac() : Sterling() {}

sterfrac(long pounds, int shillings, int pence, int eighths = 0) :

\_eighths(eighths), Sterling(pounds, shillings, pence)

{}

sterfrac(double newPounds) : Sterling(newPounds)

{

newPounds = (newPounds - (long)newPounds) \* PoundsToOldShilling;

newPounds = (newPounds - (int)newPounds) \* OldShillingToOldPennies;

\_eighths = (newPounds - (int)newPounds) \* 8;

}

sterfrac operator+(sterfrac s)

{

return sterfrac((double)\*this + (double)s);

}

sterfrac operator-(sterfrac s)

{

return sterfrac((double)\*this - (double)s);

}

sterfrac operator\*(double d)

{

return sterfrac((double)\*this \* d);

}

double operator/(sterfrac s)

{

return (double)\*this / (double)s;

}

sterfrac operator/(double d)

{

return sterfrac((double)\*this \* d);

}

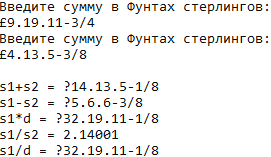
operator double()

{

return Sterling::operator double() +

\_eighths / 8.0 / OldShillingToOldPennies / PoundsToOldShilling;

}

**Вывод:**  


void get()

{

char ch;

int a, b;

Sterling::get();

cin >> ch >> a >> ch >> b;

\_eighths = a \* 8 / b;

}

void put()

{

Sterling::put();

int tnum, tden, temp;

tnum = \_eighths;

tden = 8;

while (tnum != 0) {

if (tnum < tden) {

temp = tnum;

tnum = tden;

tden = temp;

}

tnum -= tden;

}

cout << '-' << \_eighths / tden << '/' << 8 / tden;

}

sterfrac normalize(sterfrac s)

{

int a = \_eighths / 8;

if (a != 0)

{

s = (sterfrac)((Sterling)s + Sterling(0, 0, a));

s.\_eighths = \_eighths % 8;

}

return (sterfrac)Sterling::normalize(s);;

}

};

int main()

{

setlocale(LC\_ALL, "");

sterfrac s1, s2;

double d = 3.3;

s1.get();

s2.get();

cout << "\ns1+s2 = "; (s1 + s2).put();

cout << "\ns1-s2 = "; (s1 - s2).put();

cout << "\ns1\*d = "; (s1 \* d).put();

cout << "\ns1/s2 = " << (s1 / s2);

cout << "\ns1/d = "; (s1 / d).put();

return 0;

}