ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет 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 211-7210
Салов Д.К.
(Фамилия И.О.)

Дата, подпись 12.06.22
(Дата) (Подпись)

Проверил: _____ _____
(Фамилия И.О., степень, звание)
(Оценка)
Дата, подпись _____ _____
(Дата) (Подпись)

Замечания: _____
_____
_____

Москва

2021

Цель: Получить практические навыки в создании классов и их последующем использовании для наследования.

Задания:

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 c или d.

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.

```
const int LEN = 80;                //maximum length of names
////////////////////////////////////////////////////////////// class employee                    //employee class

  {    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;

        }

  };
```

6. Используя наследование, добавьте к классу, который приведён ниже, возможность для пользователя определять верхнюю и нижнюю границы массива в конструкторе. const int LIMIT = 100; //array size

```
////////////////////////////////////////////////////////////// class safearay { private: int arr[LIMIT]; public: int& operator [](int n) //note: return by reference
    {
    if( n< 0 || n>=LIMIT )
      { cout << "\nIndex out of bounds"; exit(1); }  return arr[n];
    }
  };
```

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

```
// constructors in derived class  #include <iostream> using namespace std;
////////////////////////////////////////////////////////////// class Counter {
  protected:                    //NOTE: not private
unsigned int count;             //count  public:
    Counter() : count(0)          //constructor, no args    {  }
    Counter(int c) : count(c)     //constructor, one arg
      {  }
    unsigned int get_count() const   //return count
      { return count; }
    Counter operator ++ ()        //incr count (prefix)
      { return Counter(++count); }
  };
////////////////////////////////////////////////////////////// class CountDn : public Counter
  {  public:
    CountDn() : Counter()         //constructor, no args    { }
```

```
    CountDn(int c) : Counter(c)       //constructor, 1 arg         { }
    CountDn operator -- ()            //decr count (prefix)
      { return CountDn(--count); }
};
////////////////////////////////////////
////////////////////////// int main()
  {
  CountDn c1;                         //class CountDn     CountDn c2(100);
   cout << "\nc1=" << c1.get_count();
//display    cout << "\nc2=" << c2.get_count();
//display

  ++c1; ++c1; ++c1;                   //increment c1    cout << "\nc1=" <<
c1.get_count();    //display it

  --c2; --c2;                         //decrement c2    cout << "\nc2=" << c2.get_count();    //display it

  CountDn c3 = --c2;                  //create c3 from c2    cout << "\nc3=" <<
c3.get_count();    //display c3    cout << endl;    return 0;
  }
```

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 под ходы). 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.

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.

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.

```
//multiple inheritance with employees and degrees

#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 scientist : private employee, private student  //scientist
  {   private:       int pubs;     //number of publications
public:      void getdata()        {
      employee::getdata();
      cout << "   Enter number of pubs: "; cin >> pubs;
student::getedu();          }      void putdata()
const        {          employee::putdata();          cout << "\n   Number of publications: " << pubs;
```

```
student::putedu();
        }
  };
////////////////////////////////////////////////////////////// class laborer : public employee              //laborer
  {
  };
////////////////////////////////////////////////////////////// int main()    {    manager m1;    scientist s1,
s2;    laborer l1;


  cout << endl;
  cout << "\nEnter data for manager 1";    //get data for
m1.getdata();                          //several employees

  cout << "\nEnter data for scientist 1";
s1.getdata();

  cout << "\nEnter data for scientist 2";
s2.getdata();

  cout << "\nEnter data for laborer 1";
l1.getdata();
   cout << "\nData on manager 1";          //display data for
m1.putdata();                          //several employees

  cout << "\nData on scientist 1";
s1.putdata();
```

```
   cout << "\nData on scientist 2";

s2.putdata();



   cout << "\nData on laborer 1";

l1.putdata();    cout << endl;    return 0;

   }
```

11. В различных ситуациях иногда требуется работать с двумя числами, объединенными в блок.

Например, каждая из координат экрана имеет горизонтальную составляющую (x) и вертикальную (y). 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 pop(), 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 pop(). 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().

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:

1/8 пенни — это полфартинга;
1/4 пенни — это фартинг;
3/8 пенни — это фартинг с половиной;
1/2 пенни — это полпенни;
5/8 пенни — это полфартинга плюс полпенни;
3/4 пенни — это полпенни плюс фартинг;
7/8 пенни — эго полпенни плюс фартинг с половиной.

Давайте предположим, что мы хотим добавить в класс sterling возможность пользоваться этими дробными частями пенни. Формат ввода/выво¬да 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 стерлинх'ами, 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:

Двойной щелчок для открытия кода.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>

using namespace std;



//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++4
//bud proklyat kod iz tretey zadachi //works
class publication //do not touch
{
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 //aded 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();
    }
};
class Disk : private sales, private publication {
private:
    enum etype { CD, DVD };
public:
    void getdata() {
        etype t_disk;
        char type;
        publication::getdata();
        cout << "\nDisc type (c/d): "; cin >> type;
        if (type = 'c')
            t_disk = CD;
        else t_disk = DVD;
        sales::getdata();
    }
    void putdata()const {
        publication::putdata();
        etype t_disk;
        char type;
        cout << "\nDisc type: ";
        if (t_disk = CD)
            cout << "CD";
        else
```

```
            cout << "DVD";
        sales::putdata();
    }
};

void fun4()
{
    book kniga;
    tape tp;
    Disk dk;
    kniga.getdata();
    tp.getdata();
    dk.getdata();
    cout << "\n" << "class Book: ";
    kniga.putdata();
    cout << "\n" << "class Type: ";
    tp.putdata();
    cout << "\n" << "class Disk: ";
    dk.putdata();
    cout << endl;
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++5
//works
const int LEN = 80;
class employee {
private:
    char name[LEN];
    unsigned long number;
public:
    void getdata() {
        cout << "Enter last name: ";
        cin >> name;
        cout << "Enter number: ";
        cin >> number;
    }
    void putdata() const {
        cout << "Name: " << name;
        cout << "\nNumber: " << number;
    }
};

class employee2 : private employee {
private:
    enum period { Hour, Week, Month };
    period post;
    double compensation;
public:
    void getdata() {
        employee::getdata();
        cout << "Enter compensation: ";
        cin >> compensation;
        char ch;
        cout << "Enter pay period(h/w/m): ";
        cin >> ch;
        switch (ch) {
            case 'h':
            {
                post = Hour;
                break;
            }
            case 'w':
            {
                post = Week;
```

```
                break;
            }
            case 'm':
            {
                post = Month;
                break;
            }
            default:
            {
                cout << "Error! \n\n";
                break;
            }
        }
    }
    void putdata() const {
        employee::putdata();
        cout << "\nCompensation: " << compensation;
        cout << "\nPayment type: ";
        switch (post)
        {
        case Hour: cout << "Hour"; break;
        case Week: cout << "Week"; break;
        case Month: cout << "Month"; break;
        default: { cout << "Error! \n\n"; break; }
        }
    }
};

void fun5()
{
    employee2 getInfo;
    getInfo.getdata();
    getInfo.putdata();
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++6
//works
const int LIMIT = 100;
class safearray {
private:
    int arr[LIMIT];
public:
    int& operator[](int n) {
        if (n < 0 || n >= LIMIT) {
            std::cout << "Index out of bounds";
            exit(1);
        }
        return arr[n];
    }
};

class newarray : private safearray {
private:
    int start;
    int end;
public:
    newarray(int x, int y) : start(x), end(y) {}
    int& operator[](int n) {
        return safearray::operator[](n >= end ? LIMIT : n - start);
    }
};

void fun6() {
    int start = 10;
    int end = 100;
```

```
    newarray arr(start, end);
}


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++7
//v zadanii krivoy kod //works
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 NewClass : public CountDn {
public:
    NewClass() : CountDn() {}
    NewClass(int c) : CountDn(c) {}
    NewClass operator++(int) {
        NewClass temp = *this;
        ++count;
        return temp;
    }
    NewClass operator--(int) {
        NewClass temp = *this;
        --count;
        return temp;
    }
};

void fun7()
{
    NewClass c1;
    NewClass c2(100);
    cout << "\nc1 = " << c1.get_count();
    cout << "\nc2 = " << c2.get_count();
    c1++; c2--;
    cout << "\nc1++ = " << c1.get_count();
    cout << "\nc2-- = " << c2.get_count();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++8
//nacopypastil //works
```

```
class String
{
protected:
    enum { SIZE = 80 };
    char str[SIZE];
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) > SIZE - 1) {
            int j;
            for (j = 0; j < SIZE - 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 poloska = SIZE;
        for (i = 0; i < poloska; i++)
            if (str[i] == '\0')
                poloska = i;
        for (i = 0; i < number; i++)
            substr[i] = str[i + poloska - number];
        substr[i] = '\0';
        return substr;
    }
    Pstring2 mid(int position, int number) {
        Pstring2 substr = "";
        int i;
```

```
        for (i = 0; i < SIZE - position && i < number; i++)
            substr[i] = str[i + position];
        substr[i] = '\0';
        return substr;
    }
};

void fun8()
{
    Pstring2 s1 = "TESTOVAYA STROKA";
    cout << "\nleft = ";
    s1.left(6).display();
    cout << "\nright = ";
    s1.right(6).display();
    cout << "\nmid = ";
    s1.mid(3, 5).display();
    cout << endl;
}


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++9
//works
class Publication9_1 {
private:
    char title[80];
    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 Publication9_2 : public Publication9_1 //date
{
private:
    int day, month, year;
public:
    Publication9_2() : month(0), day(0), year(0) {}
    Publication9_2(int d, int m, int y) : month(m), day(d), year(y) {}
    void getdata() {
        Publication9_1::getdata();
        cout << "Enter day, month and year (XX/XX/XXXX) \n";
        cin >> day >> month >> year;
    }
    void putdata()const {
        Publication9_1::putdata();
        cout << "\nDate publication: " << day << '/' << month << '/' << year <<
"\n";
    }
};

class book9 : private Publication9_2 {
private:
    int pages;
public:
    void getgata() {
        Publication9_2::getdata();
```

```
        cout << "Enter number of pages: ";
        cin >> pages;
    }
    void putdata()const {
        Publication9_2::putdata();
        cout << "\nPages: " << pages;
    }
};

class tape9 : private Publication9_2 {
private:
    float time;
public:
    void getdata() {
        Publication9_2::getdata();
        cout << "Enter playing time: ";
        cin >> time;
    }
    void putdata()const {
        Publication9_2::putdata();
        cout << "\nPlaying time: " << time;
    }
};

void fun9()
{
    book9 kniga;
    tape9 tp;
    kniga.getgata();
    tp.getdata();
    cout << "\n" << "first class: ";
    kniga.putdata();
    cout << "\n\n" << "second class: ";
    tp.putdata();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++10
const int LENGTH = 80;
class student
{
private:
    char school[LENGTH];
    char degree[LENGTH];
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 employee10
{
private:
    char name[LENGTH];
    unsigned long number;
public:
```

```
    void getdata()
    {
        cout << " Enter last name: ";
        cin >> name;
        cout << " Enter number: ";
        cin >> number;
    }
    void putdata() const
    {
        cout << "\n Name: " << name;
        cout << "\n Number: " << number;
    }
};

class manager : private employee10, private student
{
private:
    char title[LENGTH];
    double dues;
public:
    void getdata()
    {
        employee10::getdata();
        cout << " Enter title: ";
        cin >> title;
        cout << " Enter golf club dues: ";
        cin >> dues;
        student::getedu();
    }
    void putdata() const
    {
        employee10::putdata();
        cout << "\n Title: " << title;
        cout << "\n Golf club dues: " << dues;
        student::putedu();
    }
};

class executive : private manager
{
private:
    double prem;
    int stonks;
public:
    void getdata()
    {
        manager::getdata();
        cout << " Enter annual premium: ";
        cin >> prem;
        cout << " Enter count of stocks: ";
        cin >> stonks;
    }
    void putdata()
    {
        manager::putdata();
        cout << "\n Annual premium: " << prem;
        cout << "\n Stocks: " << stonks;
    }
};

void fun10()
{
    executive getInfo;
    getInfo.getdata();
```

```
    getInfo.putdata();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++11
//Nekorrektnaya zadacha. otsutstvuet kod.
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++12
//mnogo pasty
char UNKNOWN;
class Fraction {
protected:
    int m_numerator;
    int m_denominator;
public:
    Fraction() {}
    Fraction(int n, int d) : m_numerator(n), m_denominator(d) {}
    void get() {
        char ch;
        cin >> m_numerator >> ch >> m_denominator;
    }
    void display() const {
        cout << m_numerator << '/' << m_denominator;
    }
    Fraction operator + (Fraction value) {
        m_numerator = m_numerator * value.m_denominator + m_denominator *
value.m_numerator;
        m_denominator = m_denominator * value.m_denominator;
        return Fraction(m_numerator, m_denominator);
    }
    Fraction operator - (Fraction value) {
        m_numerator = m_numerator * value.m_denominator - m_denominator *
value.m_numerator;
        m_denominator = m_denominator * value.m_denominator;
        return Fraction(m_numerator, m_denominator);
    }
    Fraction operator * (Fraction value) {
        m_numerator = m_numerator * value.m_numerator;
        m_denominator = m_denominator * value.m_denominator;
        return Fraction(m_numerator, m_denominator);
    }
    Fraction operator / (Fraction value) {
        m_numerator = m_numerator * value.m_denominator;
        m_denominator = m_denominator * value.m_numerator;
        return Fraction(m_numerator, m_denominator);
    }
};
class divFraction : public Fraction {
public:
    divFraction() : Fraction() {}
    divFraction(int n) : Fraction(n, 8) {}
    operator double() const {
        return static_cast<double> (m_numerator) / m_denominator;
    }
};

class Sterling {
protected:
    long pounds;
    int shilling;
    int pens;
public:
    Sterling() : pounds(), shilling(), pens() {}
    Sterling(double funt) {
        pounds = static_cast<long> (funt);
        long deschast = funt - pounds;
```

```
        shilling = static_cast <int> (deschast * 20);
        pens = static_cast <int> ((funt - pounds) * 240 - shilling * 12);
    }
    Sterling(long x, int y, int z) : pounds(x), shilling(y), pens(z) {}

    void getSterling() {
        cin >> pounds >> UNKNOWN >> shilling >> UNKNOWN >> pens;
    }
    void putSterling() const {
        cout << pounds << "." << shilling << "." << pens;
    }
    operator double() const {
        return (pounds + shilling / 20.0 + pens / 240.0);
    }
    Sterling getCalculation(int sumpens) {
        long x = sumpens / 240;
        int y = sumpens % (20 * 12) / 12;
        int z = sumpens % (20 * 12) % 12;
        return Sterling(x, y, z);
    }
    Sterling operator + (Sterling value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) + (value.pounds *
240 + value.shilling * 12 + value.pens);
        return (getCalculation(sumpens));
    }
    Sterling operator - (Sterling value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) - (value.pounds *
240 + value.shilling * 12 + value.pens);
        return (getCalculation(sumpens));
    }
    Sterling operator * (double value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) * (value);
        return (getCalculation(sumpens));
    }
    Sterling operator / (Sterling value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) / (value.pounds *
240 + value.shilling * 12 +
            value.pens);
        return (getCalculation(sumpens));
    }
    Sterling operator / (double value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) / (value);
        return (getCalculation(sumpens));
    }
};

class sterfrac : public Sterling {
private:
    divFraction eighths;
public:
    sterfrac() : Sterling(), eighths(0) {}
    sterfrac(double funt) {
        pounds = funt;
        shilling = (funt - pounds) * 20;
        pens = ((funt - pounds) * 20 - shilling) * 12;
        eighths = static_cast<int>(round((((funt - pounds) * 20 - shilling) * 12
- pens) * 8));
    }
    sterfrac(long x, int y, int z, divFraction e) : Sterling(x, y, z),
eighths(e) {}
    void getSterling() {
        cout << "Enter the amount in the format '9.19.11-1/4': ";
        cin >> pounds >> UNKNOWN >> shilling >> UNKNOWN >> pens;
        cin >> UNKNOWN;
```

```
        eighths.get();
    }
    void putSterling() const {
        Sterling::putSterling();
        cout << "-";
        eighths.display();
    }
    operator double() const {
        return (Sterling::operator double() + eighths / 240.0);
    }
    sterfrac operator + (sterfrac value) const {
        return sterfrac((double)*this + (double)value);
    }
    sterfrac operator - (sterfrac value) const {
        return sterfrac((double)*this - (double)value);
    }
    sterfrac operator * (double value) const {
        return sterfrac((double)*this * value);
    }
    double operator / (sterfrac value) const {
        return (double)*this / (double)value;
    }
    sterfrac operator / (double value) const {
        return sterfrac((double)*this * value);
    }
};


void fun12()
{
    sterfrac value1, value2;
    value1.getSterling();
    value2.getSterling();
    cout << endl;
    cout << "In decimal funts: " << double(value1) << endl;
    cout << "In decimal funts: " << double(value2) << endl;
    cout << "The result of addition: ";
    (value1 + value2).putSterling();
    cout << "\n";
    cout << "The result of the difference: ";
    (value1 - value2).putSterling();
    cout << "\n";
    cout << "The result of multiplication on 1.05: ";
    (value1 * 1.05).putSterling();
    cout << "\n";
    cout << "The result of division: " << value1 / value2 << "\n";
    cout << "The result of division on 3.5: ";
    (value1 / 3.5).putSterling();
    cout << "\n";
}



int main()
{
    setlocale(LC_ALL, "Russian");
    int nomer;
    cout << "\nВведите номер задачи\n";
    cin >> nomer;
    switch (nomer)
    {
    case 4:
        fun4();
        break;
```

```
    case 5:
        fun5();
        break;
    case 6:
        fun6();
        break;
    case 7:
        fun7();
        break;
    case 8:
        fun8();
        break;
    case 9:
        fun9();
        break;
    case 10:
        fun10();
        break;
    /*case 11:
        fun11();
        break;*/
    case 12:
        fun12();
        break;
    }

    return 0;
}
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>

using namespace std;



//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++4
//bud proklyat kod iz tretey zadachi //works
class publication //do not touch
{
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 //aded 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();
    }
};
class Disk : private sales, private publication {
private:
    enum etype { CD, DVD };
public:
    void getdata() {
        etype t_disk;
        char type;
```

```
        publication::getdata();
        cout << "\nDisc type (c/d): "; cin >> type;
        if (type = 'c')
            t_disk = CD;
        else t_disk = DVD;
        sales::getdata();
    }
    void putdata()const {
        publication::putdata();
        etype t_disk;
        char type;
        cout << "\nDisc type: ";
        if (t_disk = CD)
            cout << "CD";
        else
            cout << "DVD";
        sales::putdata();
    }
};

void fun4()
{
    book kniga;
    tape tp;
    Disk dk;
    kniga.getdata();
    tp.getdata();
    dk.getdata();
    cout << "\n" << "class Book: ";
    kniga.putdata();
    cout << "\n" << "class Type: ";
    tp.putdata();
    cout << "\n" << "class Disk: ";
    dk.putdata();
    cout << endl;
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++5
//works
const int LEN = 80;
class employee {
private:
    char name[LEN];
    unsigned long number;
public:
    void getdata() {
        cout << "Enter last name: ";
        cin >> name;
        cout << "Enter number: ";
        cin >> number;
    }
    void putdata() const {
        cout << "Name: " << name;
        cout << "\nNumber: " << number;
    }
};

class employee2 : private employee {
private:
    enum period { Hour, Week, Month };
    period post;
    double compensation;
public:
    void getdata() {
```

```
        employee::getdata();
        cout << "Enter compensation: ";
        cin >> compensation;
        char ch;
        cout << "Enter pay period(h/w/m): ";
        cin >> ch;
        switch (ch) {
            case 'h':
            {
                post = Hour;
                break;
            }
            case 'w':
            {
                post = Week;
                break;
            }
            case 'm':
            {
                post = Month;
                break;
            }
            default:
            {
                cout << "Error! \n\n";
                break;
            }
        }
    }
    void putdata() const {
        employee::putdata();
        cout << "\nCompensation: " << compensation;
        cout << "\nPayment type: ";
        switch (post)
        {
        case Hour: cout << "Hour"; break;
        case Week: cout << "Week"; break;
        case Month: cout << "Month"; break;
        default: { cout << "Error! \n\n"; break; }
        }
    }
};

void fun5()
{
    employee2 getInfo;
    getInfo.getdata();
    getInfo.putdata();
}
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++6
//works
const int LIMIT = 100;
class safearray {
private:
    int arr[LIMIT];
public:
    int& operator[](int n) {
        if (n < 0 || n >= LIMIT) {
            std::cout << "Index out of bounds";
            exit(1);
        }
        return arr[n];
    }
};
```

```
class newarray : private safearray {
private:
    int start;
    int end;
public:
    newarray(int x, int y) : start(x), end(y) {}
    int& operator[](int n) {
        return safearray::operator[](n >= end ? LIMIT : n - start);
    }
};

void fun6() {
    int start = 10;
    int end = 100;
    newarray arr(start, end);
}


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++7
//v zadanii krivoy kod //works
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 NewClass : public CountDn {
public:
    NewClass() : CountDn() {}
    NewClass(int c) : CountDn(c) {}
    NewClass operator++(int) {
        NewClass temp = *this;
        ++count;
        return temp;
    }
    NewClass operator--(int) {
        NewClass temp = *this;
        --count;
        return temp;
    }
```

```
};

void fun7()
{
    NewClass c1;
    NewClass c2(100);
    cout << "\nc1 = " << c1.get_count();
    cout << "\nc2 = " << c2.get_count();
    c1++; c2--;
    cout << "\nc1++ = " << c1.get_count();
    cout << "\nc2-- = " << c2.get_count();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++8
//nacopypastil //works
class String
{
protected:
    enum { SIZE = 80 };
    char str[SIZE];
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) > SIZE - 1) {
            int j;
            for (j = 0; j < SIZE - 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 poloska = SIZE;
        for (i = 0; i < poloska; i++)
            if (str[i] == '\0')
                poloska = i;
        for (i = 0; i < number; i++)
            substr[i] = str[i + poloska - number];
        substr[i] = '\0';
        return substr;
    }
    Pstring2 mid(int position, int number) {
        Pstring2 substr = "";
        int i;
        for (i = 0; i < SIZE - position && i < number; i++)
            substr[i] = str[i + position];
        substr[i] = '\0';
        return substr;
    }
};

void fun8()
{
    Pstring2 s1 = "TESTOVAYA STROKA";
    cout << "\nleft = ";
    s1.left(6).display();
    cout << "\nright = ";
    s1.right(6).display();
    cout << "\nmid = ";
    s1.mid(3, 5).display();
    cout << endl;
}


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++9
//works
class Publication9_1 {
private:
    char title[80];
    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 Publication9_2 : public Publication9_1 //date
{
private:
    int day, month, year;
public:
    Publication9_2() : month(0), day(0), year(0) {}
    Publication9_2(int d, int m, int y) : month(m), day(d), year(y) {}
    void getdata() {
        Publication9_1::getdata();
        cout << "Enter day, month and year (XX/XX/XXXX) \n";
```

```
        cin >> day >> month >> year;
    }
    void putdata()const {
        Publication9_1::putdata();
        cout << "\nDate publication: " << day << '/' << month << '/' << year <<
"\n";
    }
};

class book9 : private Publication9_2 {
private:
    int pages;
public:
    void getgata() {
        Publication9_2::getdata();
        cout << "Enter number of pages: ";
        cin >> pages;
    }
    void putdata()const {
        Publication9_2::putdata();
        cout << "\nPages: " << pages;
    }
};

class tape9 : private Publication9_2 {
private:
    float time;
public:
    void getdata() {
        Publication9_2::getdata();
        cout << "Enter playing time: ";
        cin >> time;
    }
    void putdata()const {
        Publication9_2::putdata();
        cout << "\nPlaying time: " << time;
    }
};

void fun9()
{
    book9 kniga;
    tape9 tp;
    kniga.getgata();
    tp.getdata();
    cout << "\n" << "first class: ";
    kniga.putdata();
    cout << "\n\n" << "second class: ";
    tp.putdata();
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++10
const int LENGTH = 80;
class student
{
private:
    char school[LENGTH];
    char degree[LENGTH];
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 employee10
{
private:
    char name[LENGTH];
    unsigned long number;
public:
    void getdata()
    {
        cout << " Enter last name: ";
        cin >> name;
        cout << " Enter number: ";
        cin >> number;
    }
    void putdata() const
    {
        cout << "\n Name: " << name;
        cout << "\n Number: " << number;
    }
};

class manager : private employee10, private student
{
private:
    char title[LENGTH];
    double dues;
public:
    void getdata()
    {
        employee10::getdata();
        cout << " Enter title: ";
        cin >> title;
        cout << " Enter golf club dues: ";
        cin >> dues;
        student::getedu();
    }
    void putdata() const
    {
        employee10::putdata();
        cout << "\n Title: " << title;
        cout << "\n Golf club dues: " << dues;
        student::putedu();
    }
};

class executive : private manager
{
private:
    double prem;
    int stonks;
public:
    void getdata()
    {
        manager::getdata();
        cout << " Enter annual premium: ";
        cin >> prem;
```

```
        cout << " Enter count of stocks: ";
        cin >> stonks;
    }
    void putdata()
    {
        manager::putdata();
        cout << "\n Annual premium: " << prem;
        cout << "\n Stocks: " << stonks;
    }
};

void fun10()
{
    executive getInfo;
    getInfo.getdata();
    getInfo.putdata();
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++11
//Nekorrektnaya zadacha. otsutstvuet kod.
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++12
//mnogo pasty
char UNKNOWN;
class Fraction {
protected:
    int m_numerator;
    int m_denominator;
public:
    Fraction() {}
    Fraction(int n, int d) : m_numerator(n), m_denominator(d) {}
    void get() {
        char ch;
        cin >> m_numerator >> ch >> m_denominator;
    }
    void display() const {
        cout << m_numerator << '/' << m_denominator;
    }
    Fraction operator + (Fraction value) {
        m_numerator = m_numerator * value.m_denominator + m_denominator *
value.m_numerator;
        m_denominator = m_denominator * value.m_denominator;
        return Fraction(m_numerator, m_denominator);
    }
    Fraction operator - (Fraction value) {
        m_numerator = m_numerator * value.m_denominator - m_denominator *
value.m_numerator;
        m_denominator = m_denominator * value.m_denominator;
        return Fraction(m_numerator, m_denominator);
    }
    Fraction operator * (Fraction value) {
        m_numerator = m_numerator * value.m_numerator;
        m_denominator = m_denominator * value.m_denominator;
        return Fraction(m_numerator, m_denominator);
    }
    Fraction operator / (Fraction value) {
        m_numerator = m_numerator * value.m_denominator;
        m_denominator = m_denominator * value.m_numerator;
        return Fraction(m_numerator, m_denominator);
    }
};
class divFraction : public Fraction {
public:
    divFraction() : Fraction() {}
    divFraction(int n) : Fraction(n, 8) {}
```

```
    operator double() const {
        return static_cast<double> (m_numerator) / m_denominator;
    }
};

class Sterling {
protected:
    long pounds;
    int shilling;
    int pens;
public:
    Sterling() : pounds(), shilling(), pens() {}
    Sterling(double funt) {
        pounds = static_cast<long> (funt);
        long deschast = funt - pounds;
        shilling = static_cast <int> (deschast * 20);
        pens = static_cast <int> ((funt - pounds) * 240 - shilling * 12);
    }
    Sterling(long x, int y, int z) : pounds(x), shilling(y), pens(z) {}

    void getSterling() {
        cin >> pounds >> UNKNOWN >> shilling >> UNKNOWN >> pens;
    }
    void putSterling() const {
        cout << pounds << "." << shilling << "." << pens;
    }
    operator double() const {
        return (pounds + shilling / 20.0 + pens / 240.0);
    }
    Sterling getCalculation(int sumpens) {
        long x = sumpens / 240;
        int y = sumpens % (20 * 12) / 12;
        int z = sumpens % (20 * 12) % 12;
        return Sterling(x, y, z);
    }
    Sterling operator + (Sterling value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) + (value.pounds *
240 + value.shilling * 12 + value.pens);
        return (getCalculation(sumpens));
    }
    Sterling operator - (Sterling value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) - (value.pounds *
240 + value.shilling * 12 + value.pens);
        return (getCalculation(sumpens));
    }
    Sterling operator * (double value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) * (value);
        return (getCalculation(sumpens));
    }
    Sterling operator / (Sterling value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) / (value.pounds *
240 + value.shilling * 12 +
            value.pens);
        return (getCalculation(sumpens));
    }
    Sterling operator / (double value) {
        int sumpens = (pounds * 240 + shilling * 12 + pens) / (value);
        return (getCalculation(sumpens));
    }
};

class sterfrac : public Sterling {
private:
    divFraction eighths;
```

```
public:
    sterfrac() : Sterling(), eighths(0) {}
    sterfrac(double funt) {
        pounds = funt;
        shilling = (funt - pounds) * 20;
        pens = ((funt - pounds) * 20 - shilling) * 12;
        eighths = static_cast<int>(round((((funt - pounds) * 20 - shilling) * 12
- pens) * 8));
    }
    sterfrac(long x, int y, int z, divFraction e) : Sterling(x, y, z),
eighths(e) {}
    void getSterling() {
        cout << "Enter the amount in the format '9.19.11-1/4': ";
        cin >> pounds >> UNKNOWN >> shilling >> UNKNOWN >> pens;
        cin >> UNKNOWN;
        eighths.get();
    }
    void putSterling() const {
        Sterling::putSterling();
        cout << "-";
        eighths.display();
    }
    operator double() const {
        return (Sterling::operator double() + eighths / 240.0);
    }
    sterfrac operator + (sterfrac value) const {
        return sterfrac((double)*this + (double)value);
    }
    sterfrac operator - (sterfrac value) const {
        return sterfrac((double)*this - (double)value);
    }
    sterfrac operator * (double value) const {
        return sterfrac((double)*this * value);
    }
    double operator / (sterfrac value) const {
        return (double)*this / (double)value;
    }
    sterfrac operator / (double value) const {
        return sterfrac((double)*this * value);
    }
};


void fun12()
{
    sterfrac value1, value2;
    value1.getSterling();
    value2.getSterling();
    cout << endl;
    cout << "In decimal funts: " << double(value1) << endl;
    cout << "In decimal funts: " << double(value2) << endl;
    cout << "The result of addition: ";
    (value1 + value2).putSterling();
    cout << "\n";
    cout << "The result of the difference: ";
    (value1 - value2).putSterling();
    cout << "\n";
    cout << "The result of multiplication on 1.05: ";
    (value1 * 1.05).putSterling();
    cout << "\n";
    cout << "The result of division: " << value1 / value2 << "\n";
    cout << "The result of division on 3.5: ";
    (value1 / 3.5).putSterling();
    cout << "\n";
```

```
}


int main()
{
    setlocale(LC_ALL, "Russian");
    int nomer;
    cout << "\nВведите номер задачи\n";
    cin >> nomer;
    switch (nomer)
    {
    case 4:
        fun4();
        break;
    case 5:
        fun5();
        break;
    case 6:
        fun6();
        break;
    case 7:
        fun7();
        break;
    case 8:
        fun8();
        break;
    case 9:
        fun9();
        break;
    case 10:
        fun10();
        break;
    /*case 11:
        fun11();
        break;*/
    case 12:
        fun12();
        break;
    }

    return 0;
}
```