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 № 5

Дисциплина:

Объектно-ориентированное программирование

Тема:

Объекты и классы.

Выполнил(а): студент(ка) группы 211-7210
Салов Д.К.
(Фамилия И.О.)

Дата, подпись 17.03.22
(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.

Задания:

4. Создайте класс employee, используя задачу 4 лабораторной работы 2. Класс должен включать поле типа int для хранения номера сотрудника и поле типа float для хранения величины его оклада. Методы класса должны позволять пользователю вводить и отображать данные класса. Напишите функцию main(), которая запросит пользователя ввести данные для трех сотрудников и выведет полученную информацию на экран.

5. Взяв в качестве основы структуру из задачи 5 лабораторной работы 2, создайте класс date. Его данные должны размещаться в трех полях типа int: month, day и year. Метод класса getdate() должен принимать значение для объекта в формате 12/31/02, а метод showdate() - выводить данные на экран.

6. Расширьте содержание класса employee из задачи 4, включив в него класс date и перечисление etype (см. задачу 6 лабораторной работы 2). Объект класса date будет использоваться для хранения даты приема сотрудника на работу. Перечисление будет использовано для хранения статуса сотрудника: лаборант, секретарь, менеджер и т. д. Последние два поля данных должны быть закрытыми в определении класса employee, как и номер и оклад сотрудника. Вам будет необходимо разработать методы getemploy() и putemploy(), предназначенные соответственно для ввода и отображения информации о сотруднике. Возможно, при создании методов вам понадобится ветвление switch для работы с перечисляемым типом etype. Напишите функцию main(), которая попросит пользователя ввести данные о трех сотрудниках, а затем выведет эти данные на экран.

7. В морской навигации координаты точки измеряются в градусах и минутах широты и долготы. Например, координаты бухты Панити на о. Таити равны 149 градусов 34.8 минут восточной долготы и 17 градусов 31.5 минут южной широты. Это записывается как 149°34.8' W, 17°31.5' S. Один градус равен 60 минутам (устаревшая система также делила одну минуту на 60 секунд, но сейчас минуту делят на обычные десятичные доли). Долгота измеряется величиной от 0 до 180 градусов восточнее или западнее Гринвича. Широта принимает значения от 0 до 90 градусов севернее или южнее экватора.

Создайте класс angle, включающий следующие три поля: типа int для числа градусов, типа float для числа минут и типа char для указания направления (N, S, E или W). Объект этого класса может содержать значение как широты, так и долготы. Создайте метод, позволяющий ввести координату точки, направление, в котором она измеряется, и метод, выводящий на экран значение этой

координаты, например 179°59.9' E. Кроме того, напишите конструктор, принимающий три аргумента. Напишите функцию mainQ, которая сначала создает переменную с помощью трехаргументного конструктора и выводит ее значение на экран, а затем циклически запрашивает пользователя ввести значение координаты и отображает введенное значение на экране. Для вывода символа градусов (°) можно воспользоваться символьной константой '\xF8'.

8. Создайте класс, одно из полей которого хранит «порядковый номер» объекта, то есть для первого созданного объекта значение этого поля равно 1, для второго созданного объекта значение равно 2 и т. д.

Для того чтобы создать такое поле, вам необходимо иметь еще одно поле, в которое будет записываться количество созданных объектов класса (это означает, что последнее поле должно относиться не к отдельным объектам класса, а ко всему классу в целом.

Вспомните, какое ключевое слово необходимо при описании такого поля.). Каждый раз при создании нового объекта конструктор может получить значение этого поля и в соответствии с ним назначить объекту индивидуальный порядковый номер.

В класс следует включить метод, который будет выводить на экран порядковый номер объекта. Создайте функцию main(), в которой будут созданы три объекта, и каждый объект выведет на экран свой порядковый номер, например: Мой порядковый номер: 2 и т. п.

9. На основе структуры fraction из упражнения 8 главы 4 создайте класс fraction. Данные класса должны быть представлены двумя полями: числителем и знаменателем. Методы класса должны получать от пользователя значения числителя и знаменателя дроби в форме 3/5 и выводить значение дроби в этом же формате. Кроме того, должен быть разработан метод, складывающий значения двух дробей. Напишите функцию main(), которая циклически запрашивает у пользователя ввод пары дробей, затем складывает их и выводит результат на экран. После каждой такой операции программа должна спрашивать пользователя, следует ли продолжать цикл.

10. Создайте класс с именем ship, который будет содержать данные об учетном номере корабля и координатах его расположения. Для задания номера корабля следует использовать механизм, аналогичный описанному в упражнении 8. Для хранения координат используйте два поля типа angle (см. упражнение 7). Разработайте метод, который будет сохранять в объекте данные о корабле, вводимые пользователем, и метод, выводящий данные о корабле на экран. Напишите функцию mainQ, создающую три объекта класса ship, затем запрашивающую ввод пользователем информации о каждом из кораблей и выводящую на экран всю полученную информацию.

11. Модифицируйте калькулятор, созданный в упражнении 12 главы 5 так, чтобы вместо структуры fraction использовался одноименный класс. Класс должен содержать методы для ввода и вывода данных объектов, а также для выполнения арифметических операций. Кроме того, необходимо включить в состав класса функцию, приводящую дробь к несократимому виду. Функция должна находить наибольший общий делитель числителя и знаменателя и делить числитель и знаменатель на это значение. Код функции, названной lowterms(), приведен ниже:

```
void fraction::lowterms() //Сокращение дроби

{

        long tnum, tden, temp,gcd;          tnum = labs(num); //
Используем неотрицательные значения

        tden = labs(den); // Нужен cmath          if(tden == 0) { //
Проверка знаменателя

                cout << “Недопустимый знаменатель”;  exit(1);

        } else if(tnum == 0)

{       num = 0;          den = 1;

                return;

}

// Нахождение наибольшего общего делителя   while(tnum != 0) {

                        if(tnum < tden) { // Если числитель больше

знаменателя, меняем их местами.          temp = tnum;    tnum = tden;

                tden = temp;

        }

        tnum = tnum – tden;

}

gcd = tden;// Делим числитель и знаменатель на НОД.  num = num / gcd;
```

den = den / gcd;

}

Можно вызывать данную функцию в конце каждого метода, выполняющего арифметическую операцию, либо непосредственно перед выводом на экран результата. Кроме перечисленных методов, вы можете включить в класс конструктор с двумя аргументами, что также будет полезно.

12. Используйте преимущество ООП, заключающееся в том, что однажды созданный класс можно помещать в другие программы. Создайте новую программу, которая будет включать класс fraction, созданный в упражнении 11. Программа должна выводить аналог целочисленной таблицы умножения для дробей. Пользователь вводит знаменатель, а программа должна подобрать всевозможные целые значения числителя так, чтобы значения получаемых дробей находились между 0 и 1. Дроби из получившегося таким образом набора перемножаются друг с другом во всевозможных комбинациях, в результате чего получается таблица следующего вида (для знаменателя, равного 6):

1/61/31/22/35/6

1/61/361/181/121/95/36

1/31/181/91/62/95/18

1/21/121/61/41/35/12

2/31/92/91/34/95/9

5/65/365/185/125/925/36

Код:

```
#include <iostream>
#include <iomanip>
using namespace std;


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++4
class Employee{
private:
          int number;
          float salary;
public:
          void get_emp()
          {
                    cout << "Number:" << endl;
                    cin >> number;
```

```
                    cout << "Salary:" << endl;
                    cin >> salary;
          }
          void display()const
          {
                    cout << "employee " << number << ", salary " << salary << "$" << endl;
          };
};
void fun4() {
          Employee emp1, emp2, emp3;
          emp1.get_emp();
          emp2.get_emp();
          emp3.get_emp();
          emp1.display();
          emp2.display();
          emp3.display();
          system("pause");
}
//-------------------------------------------------------------------------4


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++5

class Date{
private:
          int day, month, year;
public:
          void getdate()
          {
                    char ch = '/';
                    cout << "Enter date (day/month/year) " << endl;
                    cin >> day >> ch >> month >> ch >> year;
          }
          void showdate()const
          {
                    char ch = '/';
                    cout << "You entered this date: " << day << ch << month << ch << year << endl;
          };
};
void fun5() {
          Date date;
          date.getdate();
          date.showdate();
}
//-------------------------------------------------------------------------5


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++6
class Date6
{
private:
public:
          int day, month, year;
};
class Employee6
{
private:
          int number;
          float salary;
          Date6 date;
          enum etype { laborer, secretary, manager, accountant, executive, researcher };
          etype post;
          char letter;
public:
          void get_empl()
          {
                    char ch;
                    cout << "Number: " << endl;
                    cin >> number;
                    cout << "Salary: " << endl;
                    cin >> salary;
                    cout << "Date of the employment (day/month/year) " << endl;
                    cin >> date.day >> ch >> date.month >> ch >> date.year;
                    cout << "Enter first letter of post (laborer, secretary, manager, accountant, executive, researcher)" << endl;
                    cin >> letter;
                    switch (letter)
                    {
                    case 'l': post = laborer; break;
                    case 's': post = secretary; break;
                    case 'm': post = manager; break;
                    case 'a': post = accountant; break;
                    case 'e': post = executive; break;
                    case 'r': post = researcher; break;
                    }
          }
          void put_empl()const
          {
                    cout << "employee " << number << ", salary " << salary << "$," << " date of employment " << date.day << "/" << date.month <<
"/" << date.year << ", post ";
                    switch (post)
                    {
                    case laborer: cout << "laborer" << endl; break;
                    case secretary: cout << "secretary" << endl; break;
                    case manager: cout << "manager" << endl; break;
                    case accountant: cout << "accountant" << endl; break;
                    case executive: cout << "executive" << endl; break;
                    case researcher: cout << "researcher" << endl; break;
                    }
          };
};
void fun6() {
          Employee6 empl_1, empl_2, empl_3;
          empl_1.get_empl();
```

```
          empl_2.get_empl();
          empl_3.get_empl();
          empl_1.put_empl();
          empl_2.put_empl();
          empl_3.put_empl();
}
//-------------------------------------------------------------------------6

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++7
class Angle
{
private:
          int degree;
          float min;
          char dir;
public:
          Angle(int degree, float min, char dir)
          {}
          void get_dir()
          {
                    cout << "Enter degree" << endl; cin >> degree;
                    cout << "Enter minutes" << endl; cin >> min;
                    cout << "Enter direction (N, S, E, W)" << endl; cin >> dir;
          }
          void show_dir()
          {
                    cout << degree << '\xF8' << min << "'" << dir << endl;
          }
};

void mainQ() {
          Angle angl2(179, 59.9, 'E');
          char ch;
          do{
                    angl2.get_dir();
                    angl2.show_dir();
                    cout << "repeat? (y/n)" << endl;
                    cin >> ch;
          } while (ch != 'n');
}

void fun7() {
          Angle angl1(179, 59.9, 'E');
          char ch;
          angl1.get_dir();
          angl1.show_dir();
          mainQ();
}
//-------------------------------------------------------------------------7

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++8
class Amount
{
private:
          int number;
          static int amount;
public:
          void disp_num()
          {
                    amount++;
                    number = amount;
                    cout << number << endl;
          }
};
int Amount::amount = 0;

void fun8() {
          Amount obj1, obj2, obj3;
          obj1.disp_num();
          obj2.disp_num();
          obj3.disp_num();
          system("pause");
}
//-------------------------------------------------------------------------8

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++9
class Fraction
{
private:
          int numerator, denominator;
public:
          char ch;
          void get_fr()
          {
                    cout << "enter fraction" << endl;
                    cin >> numerator >> ch >> denominator;
          }
          void add_fr(Fraction f1, Fraction f2)
          {
                    numerator = f1.numerator * f2.denominator + f1.denominator * f2.numerator;
                    denominator = f1.denominator * f2.denominator;
          }
          void disp_fr()
          {
                    cout << numerator << "/" << denominator << endl;
          }
};
void fun9() {
          Fraction fr1, fr2, fr_sum;
          char ch;
          do{
                    fr1.get_fr();
                    fr2.get_fr();
```

```
                    fr_sum.add_fr(fr1, fr2);
                    fr_sum.disp_fr();
                    cout << "repeat? (y/n)" << endl; cin >> ch;
          } while (ch != 'n');
}
//-------------------------------------------------------------------------9

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++10
class Ship
{
private:
          int number;
          static int amount;
          int degree;
          float min;
          char dir;
public:
          void get_sh()
          {
                    amount++;
                    number = amount;
                    cout << "Enter degree" << endl; cin >> degree;
                    cout << "Enter minutes" << endl; cin >> min;
                    cout << "Enter direction (N, S, E, W)" << endl; cin >> dir;
          }
          void disp_sh()
          {
                    cout << "Number: " << number << ", Direction: " << degree << '\xF8' << min << "'" << dir << endl;
          }
};
int Ship::amount = 0;

void fun10() {
          Ship ship1, ship2, ship3;
          ship1.get_sh();
          ship2.get_sh();
          ship3.get_sh();
          ship1.disp_sh();
          ship2.disp_sh();
          ship3.disp_sh();
}
//-------------------------------------------------------------------------10

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++11
class Fraction11
{
private:
          int numerator, denominator;
public:
          void get_fr()
          {
                    char ch;
                    cout << "enter fraction" << endl;
                    cin >> numerator >> ch >> denominator;
          }
          void add_fr(Fraction11 f1, Fraction11 f2)
          {
                    numerator = f1.numerator * f2.denominator + f1.denominator * f2.numerator;
                    denominator = f1.denominator * f2.denominator;
          }
          void sub_fr(Fraction11 f1, Fraction11 f2)
          {
                    numerator = f1.numerator * f2.denominator - f1.denominator * f2.numerator;
                    denominator = f1.denominator * f2.denominator;
          }
          void mul_fr(Fraction11 f1, Fraction11 f2)
          {
                    numerator = f1.numerator * f2.denominator;
                    denominator = f1.denominator * f2.denominator;
          }
          void div_fr(Fraction11 f1, Fraction11 f2)
          {
                    numerator = f1.numerator * f2.denominator;
                    denominator = f1.denominator * f2.numerator;
          }
          void lowterms()
          {
                    long tnum, tden, temp, gcd;
                    tnum = labs(numerator);            // используем неотрицательные
                    tden = labs(denominator);            // значения (нужен cmath)
                    if (tden == 0)              // проверка знаменателя на 0
                    {
                              cout << "Недопустимый знаменатель!"; exit(1);
                    }
                    else if (tnum == 0)         // проверка числителя на 0
                    {
                              numerator = 0; denominator = 1; return;
                    }
                    // нахождение наибольшего общего делителя
                    while (tnum != 0)
                    {
                              if (tnum < tden)          // если числитель больше знаменателя,
                              {
                                        temp = tnum; tnum = tden; tden = temp;
                              } //меняем их местами
                              tnum = tnum - tden;        // вычитание
                    }
                    gcd = tden;                  // делим числитель и знаменатель на
                    numerator = numerator / gcd;             // полученный наибольший общий делитель
                    denominator = denominator / gcd;
          }
          void disp_fr()const
          {
```

```
                    cout << numerator << "/" << denominator << endl;
          }
};

void fun11() {
          char zn;
          do
          {
                    Fraction11 fr1, fr2, fr_sum;
                    fr1.get_fr();
                    cout << "Enter sign" << endl; cin >> zn;
                    fr2.get_fr();
                    switch (zn)
                    {
                    case '+': fr_sum.add_fr(fr1, fr2); break;
                    case '-': fr_sum.sub_fr(fr1, fr2); break;
                    case '*': fr_sum.mul_fr(fr1, fr2); break;
                    case '/': fr_sum.div_fr(fr1, fr2); break;
                    }
                    fr_sum.lowterms();
                    fr_sum.disp_fr();
                    cout << "repeat? (y/n)" << endl; cin >> zn;
          } while (zn != 'n');
}
//------------------------------------------------------------------------11

int main(){
          int nomer;
          cout << "\nEnter # of exercise \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;
          }
          return 0;
}
```

Код программки для 12й задачи (внутри класса внесены изменения в область видимости двух переменных):

```
#include <iostream>
#include <iomanip>
using namespace std;

class Fraction
{
private:

public:
    int numerator, denominator;
    Fraction() :numerator(0), denominator(0)
    {}
    void mul_fr(Fraction f1, Fraction f2)
    {
        numerator = f1.numerator * f2.numerator;
        denominator = f1.denominator * f2.denominator;
    }
    void lowterms()
    {
        long tnum, tden, temp, gcd;
        tnum = labs(numerator);
        tden = labs(denominator);
        if (tden == 0)
        {
            cout << "Недопустимый знаменатель!"; exit(1);
        }
        else if (tnum == 0)
        {
            numerator = 0; denominator = 1; return;
        }
        while (tnum != 0)
        {
            if (tnum < tden)
            {
                temp = tnum; tnum = tden; tden = temp;
            }
            tnum = tnum - tden;
        }
```

```
        gcd = tden;
        numerator = numerator / gcd;
        denominator = denominator / gcd;
    }
    void disp_fr()const
    {
        cout << setw(5) << numerator << "/" << denominator;
    }
};
int main()
{
    int num;
    Fraction fr1, fr2, fr3;
    fr1.denominator = fr2.denominator = num;
    cout << "enter denominator" << endl; cin >> num;
    for (int j = 1; j < num; j++)
    {
        fr1.numerator = j;
        fr1.denominator = num;
        fr1.lowterms();
        fr1.disp_fr();
    }
    cout << endl;
    for (int i = 1; i < num; i++)
    {
        fr2.numerator = i;
        fr2.denominator = num;
        fr2.lowterms();
        fr2.disp_fr();

        for (int j = 1; j < num; j++)
        {
            fr1.numerator = j;
            fr1.denominator = num;
            fr3.mul_fr(fr1, fr2);
            fr3.lowterms();
            fr3.disp_fr();
        }
        cout << endl;
    }
    system("pause");
    return 0;
}
```