Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

ДРУЖЕСТВЕННЫЕ ФУНКЦИИ. НАСЛЕДОВАНИЕ КЛАССОВ. МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ

Отчет по лабораторной работе по дисциплине «Объектно-ориентированное программирование»

Студент группы 549

 Баулин С.К.

«__» _____ 2020 г.

Кандидат физико-математических наук, доцент кафедры ЭМИС

_____ _____ Шельмина Е. А.

оценка «__» _____ 2020 г.

Томск 2020

Лабораторная работа №7

Дружественные функции. Наследование классов. Множественное наследование.

Цель работы: освоить и применить на практике дружественные функции, наследование классов, множественное наследование.

Теоретические сведения

Дружественной функцией класса называется функция, которая, не являясь его компонентом, имеет доступ к его защищённым и собственным компонентам. Для реализации прав друга функция должна быть описана в теле класса со спецификатором friend («друг»).

Функция frnd_put() описана в классе charlocus как дружественная функция и определена обычным образом как глобальная функция (вне класса, без ука-зания его имени, без операции :: и без спецификатора friend). Как дружес-твенная, она получает доступ к защищённым (protected) дан-ным класса и изменяет значение символа того объекта, адрес которого будет передан ей как значение первого параметра.

Функция может быть дружественной по отношению к нескольким классам:

- class CL2;
- class CL1 friend void ff(CL1,CL2); ...
- class CL2 friend void ff(CL1,CL2); ...
- void ff(...) тело функции.

Наследование

Наследование - это механизм получения нового класса на основе уже существующего. Существующий класс может быть дополнен или изменен для создания нового класса.

Существующие классы называются базовыми, а новые – производными. Производный класс наследует описание базового класса; затем он может быть

изменен добавлением новых членов, изменением существующих функций-членов и изменением прав доступа. С помощью наследования может быть создана иерархия классов, которые совместно используют код и интерфейсы.

Наследуемые компоненты не перемещаются в производный класс, а остаются в базовых классах.

В иерархии производный объект наследует разрешенные для наследования компоненты всех базовых объектов (public, protected).

Допускается множественное наследование – возможность для некоторого класса наследовать компоненты нескольких никак не связанных между собой базовых классов. В иерархии классов соглашение относительно доступности компонентов класса следующее:

private – член класса может использоваться только функциями – членами данного класса и функциями – “друзьями” своего класса. В производном классе он недоступен.

protected – то же, что и private, но дополнительно член класса с данным атрибутом доступа может использоваться функциями-членами и функциями – “друзьями” классов, производных от данного.

public – член класса может использоваться любой функцией, которая является членом данного или производного класса, а также к public - членам возможен доступ извне через имя объекта.

Следует иметь в виду, что объявление friend не является атрибутом доступа и не наследуется.

Синтаксис определения производного класса:

class имя_класса : список_базовых_классов

{список_компонентов_класса};

В производном классе унаследованные компоненты получают статус доступа private, если новый класс определен с помощью ключевого слова class, и статус public, если с помощью struct.

Явно изменить умалчиваемый статус доступа при наследовании можно с помощью атрибутов доступа – private, protected и public, которые указываются непосредственно перед именами базовых классов.

Конструкторы и деструкторы производных классов

Поскольку конструкторы не наследуются, при создании производного класса наследуемые им данные-члены должны инициализироваться конструктором базового класса. Конструктор базового класса вызывается автоматически и выполняется до конструктора производного класса. Параметры конструктора базового класса указываются в определении конструктора производного класса. Таким образом происходит передача аргументов от конструктора производного класса конструктору базового класса.

Например.

```
class Basis
{ int a,b;
public:
Basis(int x,int y){a=x;b=y;}
};
class Inherit:public Basis
{int sum;
public:
Inherit(int x,int y, int s):Basis(x,y){sum=s;}
};
```

Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты-объекты (если они имеются), а потом сам производный класс. Таким образом, объект производного класса содержит в качестве подобъекта объект базового класса.

Уничтожаются объекты в обратном порядке: сначала производный, потом его компоненты-объекты, а потом базовый объект.

Таким образом, порядок уничтожения объекта противоположен по отношению к порядку его конструирования.

Задания для самостоятельной работы.

Задание 1. Реализовать дружественные функции для работы с объектами классов.

Даны два массива (классы множества чисел). Написать дружественную функцию, сортирующую эти массивы по возрастанию. Скриншоты с кодом программы и результатом приведены на рисунках 1.1 – 1.4.

Задание 2. Наследование. Требуется создать базовый класс и определить общие и специфические методы для данного класса. Создать производные классы, в которые добавить свойства и методы. Часть методов переопределить. Создать массив объектов базового класса и заполнить объектами производных классов. Предусмотреть передачу аргументов конструкторам базового класса, использовать виртуальные и перегруженные функции.

Создать базовый класс «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость перевозки для указанных городов и расстояний.

Скриншоты с кодом программы и результатом приведены на рисунках 2.1 – 2.4.

Задание 3. Множественное наследование. Необходимо построить иерархию классов согласно схеме наследования, приведенной в варианте задания. Каждый класс должен содержать инициализирующий конструктор и функцию show для вывода значений. Функция main должна иллюстрировать иерархию наследования.

Скриншоты с заданием, кодом программы и результатом приведены на рисунках 3.1 – 3.5.

```
1  > #include <iostream>···
6    using namespace std;
7    #define MAX 100
8
9    class IntMass;
10   class FloatMass;
11
     You, 2 months ago | 1 author (You)
12   class IntMass
13   {
14   private:
15       int data[MAX];
16       int size;
17
18   public:
19       IntMass(int msize)
20       {
21           cout << "Массив Int: ";
22           for (int i = 0; i < msize; i++)
23           {
24               this->data[i] = rand() % 100;
25               cout << this->data[i] << " ";
26           }
27           cout << endl;
28           this->size = msize;
29       };
30       friend void sort(IntMass &massInt, FloatMass &massFloat);
31   };
```

Рисунок 1.1 – Скриншот кода программы

```
32  class FloatMass
33  {
34  private:
35      float data[MAX];
36      int size;
37
38  public:
39      FloatMass(int msize)
40      {
41          cout << "Массив float: ";
42          for (int i = 0; i < msize; i++)
43          {
44              this->data[i] = (float)(rand() % 1000 + 1) / 10;
45              cout << this->data[i] << " ";
46          }
47          cout << endl;
48          this->size = msize;
49      }
50      friend void sort(IntMass &massInt, FloatMass &massFloat);
51  };
52
```

Рисунок 1.2 – скриншот кода программы

```
53  void sort(IntMass &massInt, FloatMass &massFloat)
54  {
55      int temp;
56      for (int i = 0; i < massInt.size; i++)
57      {
58          for (int j = 0; j < massInt.size - 1; j++)
59          {
60              if (massInt.data[j] > massInt.data[j + 1])
61              {
62                  temp = massInt.data[j];
63                  massInt.data[j] = massInt.data[j + 1];
64                  massInt.data[j + 1] = temp;
65              }
66          }
67      }
68      float temp2;
69      for (int i = 0; i < massFloat.size; i++)
70      {
71          for (int j = 0; j < massFloat.size - 1; j++)
72          {
73              if (massFloat.data[j] > massFloat.data[j + 1])
74              {
75                  temp2 = massFloat.data[j];
76                  massFloat.data[j] = massFloat.data[j + 1];
77                  massFloat.data[j + 1] = temp2;
78              }
79          }
80      }
81  }
82  int main()
83  {
84      setlocale(LC_ALL, "65001");
85      srand(time(NULL));
86      IntMass a(10);
87      FloatMass b(8);
88      sort(a, b);
89
90      return 0;
91  }
```

Рисунок 1.3 – Скриншот кода программы

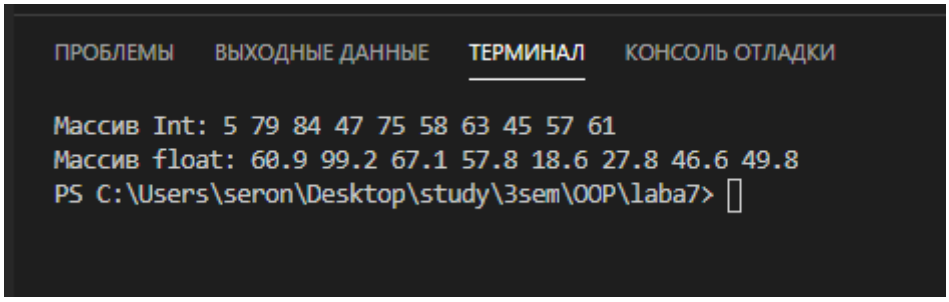

Рисунок 1.4 – Скриншот результата работы программы

```
7   class TransportCompany
8   {
9   protected:
10      double speed, price, distance;
11      string type, city1, city2;
12
13  public:
        You, a month ago | 1 author (You)
14      TransportCompany()
15      {
16          speed = price = 0;
17          type = "";
18      }
19      virtual void Write()
20      {
21          cout << "======================" << endl;
22          cout << type << endl;
23          cout << "Скорость " << speed << " км/ч" << endl;
24          cout << "Плата " << price << " руб/ч" << endl;
25      }
26      float Time()
        You, a month ago | 1 author (You)
27      {
28          float time = distance / speed;
29          return int(time * 100 + 0.5) / 100.0;
30      }
31      float Cost()
32      {
33          return Time() * price;
34      };
35
36      virtual void CheckCount()
37      {
38          cout << "======================" << endl;
39          cout << type << endl;
            You, a month ago | 1 author (You)
40          cout << "Скорость " << speed << " км/ч" << endl;
41          cout << "Плата " << price << " руб/ч" << endl;
42          cout << "Из города " << city1 << " в город " << city2 << endl;
43          cout << "Путь " << distance << " км." << endl;
44          cout << "Займёт времени " << Time() << " ч." << endl;
45          cout << "Стоимость " << Cost() << " руб." << endl;
46      }
47  };
```

Рисунок 2.1 – Скриншот кода программы

```
49  class Airplane : public TransportCompany
50  {
51
52  public:
53      Airplane()
54      {
55          speed = 800.00;
56          price = 2800;
57          type = "Airplane";
58      }
59      Airplane(string c1, string c2, double d)
60      {
61          speed = 800.00;
62          price = 2800;
63          type = "Airplane";
64          city1 = c1;
65          city2 = c2;
66          distance = d;
67      }
68  };
69
70  class Train : public TransportCompany
71  {
72  public:
73      Train()
74      {
75          speed = 220.00;
76          price = 1700;
77          type = "Train";
78      }
79      Train(string c1, string c2, double d)
80      {
81          speed = 220.00;
82          price = 1700;
83          type = "Train";
84          city1 = c1;
85          city2 = c2;
86          distance = d;
87      }
88  };
```

Рисунок 2.2 – Скриншот кода программы

```
90  class Car : public TransportCompany
91  {
92  public:
93      Car()
94      {
95          speed = 80.00;
96          price = 960;
97          type = "Car";
98      }
99      Car(string c1, string c2, double d)
100     {
101         speed = 80.00;
102         price = 960;
103         type = "Car";
104         city1 = c1;
105         city2 = c2;
106         distance = d;
107     }
108 };
109
110 int main()
111 {
112     setlocale(LC_ALL, "65001");
113     system("chcp 65001");
114     system("cls");
115
116     TransportCompany *arr[3]{
117         arr[0] = new Airplane("Москва", "Париж", 2487),
118         arr[1] = new Train("Москва", "Париж", 2509),
119         arr[2] = new Car("Москва", "Париж", 2827)};
120     for (int i = 0; i < 3; i++)
121     {
122         arr[i]->CheckCount();
123         cout << endl;
124     }
125
126     return 0;
127 }
```

Рисунок 2.3 – Скриншот кода программы

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  КОНСОЛЬ ОТЛАДКИ

========================
Airplane
Скорость 800 км/ч
Плата 2800 руб/ч
Из города Москва в город Париж
Путь 2487 км.
Займёт времени 3.11 ч.
Стоимость 8708 руб.

========================
Train
Скорость 220 км/ч
Плата 1700 руб/ч
Из города Москва в город Париж
Путь 2509 км.
Займёт времени 11.4 ч.
Стоимость 19380 руб.

========================
Car
Скорость 80 км/ч
Плата 960 руб/ч
Из города Москва в город Париж
Путь 2827 км.
Займёт времени 35.34 ч.
Стоимость 33926.4 руб.
```

Рисунок 2.4 – Скриншот результата работы программы

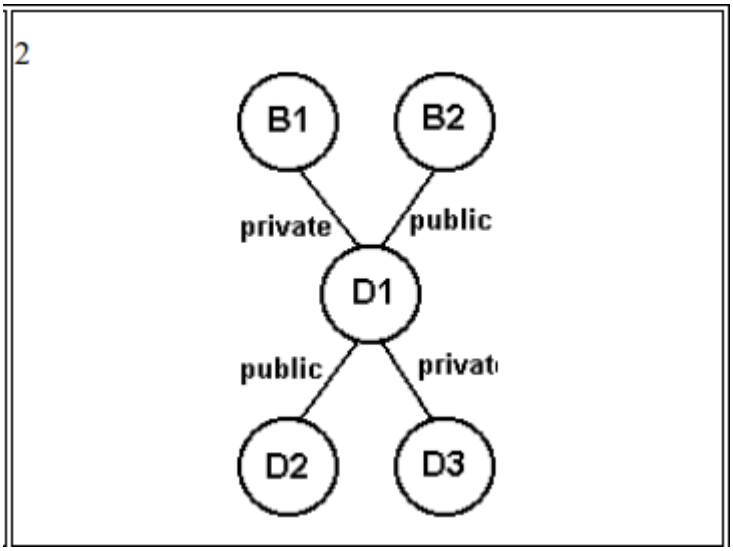

Рисунок 3.1 – Скриншот задания 3

```
 7  class B1
 8  {
 9  private:
10      int a;
11
12  public:
13      B1() { a = 0; }
14      B1(int x) { a = x; }
15      void show_B1() { cout << "B1 = " << a << endl; }
16  };
    You, seconds ago | 1 author (You)
17  class B2
18  {
19  private:
20      int b;
21
22  public:
23      B2() { b = 0; }
24      B2(int x) { b = x; }
25      void show_B2() { cout << "B2 = " << b << endl; }
26  };
    You, seconds ago | 1 author (You)
27  class D1 : private B1, public B2
28  {
29  private:
30      int c;
31
32  public:
33      D1() { c = 0; }
34      D1(int x, int y, int z) : B1(y), B2(z) { c = x; }
35      void show_D1()
36      {
37          cout << "D1 = " << c << endl;
38          show_B1();
39          show_B2();
40      }
41  };
```

Рисунок 3.2 – Скриншот кода программы

```
43  class D2 : public D1
44  {
45  private:
46      int d;
47
48  public:
49      D2() { d = 0; }
50      D2(int x, int y, int z, int i) : D1(y, z, i) { d = x; }
51      void show_D2()
52      {
53          cout << "D2 = " << d << endl;
54          show_D1();
55      }
56  };
57
    You, seconds ago | 1 author (You)
58  class D3 : public D1
59  {
60  private:
61      int d;
62
63  public:
64      D3() { d = 0; }
65      D3(int x, int y, int z, int i) : D1(y, z, i) { d = x; }
66      void show_D3()
67      {
68          cout << "D3 = " << d << endl;
69          show_D1();
70      }
71  };
```

Рисунок 3.3 – Скриншот кода программы

```
73  main()
74  {
75      setlocale(LC_ALL, "65001");
76      system("chcp 65001");
77      system("cls");
78
79      D2 temp1(1, 2, 3, 4);
80      D3 temp2(100, 200, 300, 400);
81      cout << "D3 temp1(1, 2, 3, 4);\nD3 temp2(100, 200, 300, 400);\n";
82      cout << "Следуя иерархии класса D2: \n";
83      temp1.show_D2();
84      cout << "Следуя иерархии класса D3: \n";
85      temp2.show_D3();
86
87      return 0;
88  }
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ ТЕРМИНАЛ КОНСОЛЬ ОТЛАДКИ

```
D3 temp1(1, 2, 3, 4);
D3 temp2(100, 200, 300, 400);
Следуя иерархии класса D2:
D2 = 1
D1 = 2
B1 = 3
B2 = 4
Следуя иерархии класса D3:
D3 = 100
D1 = 200
B1 = 300
B2 = 400
```

Рисунок 3.4 – Скриншот результата работы программы

Вывод: освоены и применены на практике дружественные функции, наследование классов, множественное наследование.