Язык программирования C++

Лекция 6

Структуры

Объектно-ориентированный подход:

- классы

Структуры

Массив – совокупность данных одного типа, например: float x[80];
Структура – совокупность данных различных типов, сгруппированная под единым именем:

```
struct book {                    //шаблон представления нового типа данных book
    char title[100];
    char author [100];
    int year;
};
```

Объявление переменной для структуры типа book:

```
struct book lib;
                    lib.year
```

Массивы структур:

```
struct book lib[70];
                    lib[37].year
```

Указатель на структуру:

```
struct book *lib;
                    lib->year
```

Ключевое слово struct в объявлениях переменных, массивов и указателей может быть опущено.

book становится новым типом данных, таким же как int или float

Классы

При решении научных и инженерных задач мы редко впрямую оперируем внутренними компьютерными типами данных: битами, байтами, целыми числами или числами с плавающей точкой

Например, разрабатывая программу восстановления треков заряженных частиц в детекторе, приходится иметь дело с такими объектами, как:

- треки
- точки в пространстве
- массивы точек
- элементы трекового детектора
- цилиндры
- слои
- и т.д.

C++ вводит механизм **классов**, который позволяет не только определять новые типы сложных данных, но и операции над ними

Типичный пример: Class Library for High Energy Physics (CLHEP)

Класс Hep3Vector

Упрощенный пример из библиотеки CLHEP: класс 3х векторов

```
class Hep3Vector {
public:
        Hep3Vector();
        Hep3Vector(double x, double y, double z);
        double x();
        double y();
        double z();
        double phi();
        double cosTheta();
        double mag();
        // much more not shown
private:
        double dx, dy, dz;
};
```

- это объявление класса, которое обычно помещается в заголовочном файле
- объявление начинается с ключевого слова **class**
- объявление класса заключено в { }
- объявление содержит элементы данных (data members) и методы класса (member functions)
- имя класса становится именем этого типа данных, который ввел программист

Использование объекта класса

```
#include <iostream>
#include <CLHEP/ThreeVector.h>
using namespace std;

int main () {
    double x, y, z;

    while ( cin >> x >> y >> z ) {
        Hep3Vector aVec(x, y, z);

        cout << “r: “ << aVec.mag();
        cout << “  phi: “ << aVec.phi();
        cout << “  cos(theta): “ << aVec.cosTheta() << endl;
    }
    return 0;
}
```

- оператор Hep3Vector aVec(x, y, z); определяет aVec – объект типа Hep3Vector и инициализирует его
- aVec.mag() вызывает метод mag() этого объекта
- “ . ” – оператор доступа к членам класса
- если используется не сам объект, а указатель на него, то доступ к членам класса осуществляется оператором “->”

Члены класса (элементы данных)

```
class Hep3Vector {
public:
   //  member functions

private:
   double dx, dy, dz;
};
```

- класс Hep3Vector содержит 3 элемента данных
- они декларируются обычным образом, за исключением запрета на инициализацию, т.е. на присвоение начальных значений
- каждый объект класса Hep3Vector будет иметь свои собственные 3 элемента данных

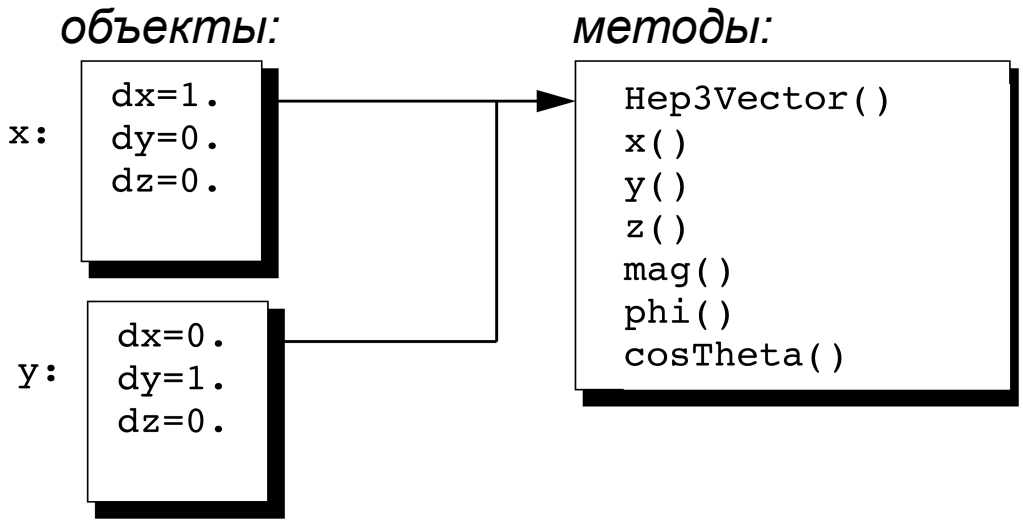
```
Hep3Vector x(1.0, 0.0, 0.0);
Hep3Vector y(0.0, 1.0, 0.0);
Hep3Vector z(0.0, 0.0, 1.0);
```

- Hep3Vector – это тип данных
- значение (или состояние) объекта типа Hep3Vector определяется значениями его элементов данных
- размер объекта типа Hep3Vector в памяти будет составлять 3*sizeof(double)

Расположение в памяти

```
Hep3Vector x(1.0, 0.0, 0.0);
Hep3Vector y(0.0, 1.0, 0.0);
```

В памяти компьютера имеем:



- каждый объект – это экземпляр (instance) класса
- каждый объект имеет свои собственные элементы данных
- методы класса являются общими для всех экземпляров класса

Скрытые члены класса

```
class Hep3Vector {
private:
        double dx, dy, dz;

public:
        double mag();
        double x();
        double dummy;
        //  more member functions
};
```

Правильно

```
Hep3Vector x(1.0, 0.0, 0.0);
cout << x.dummy;
```

Ошибка

```
Hep3Vector x(1.0, 0.0, 0.0);
cout << x.dx;
```

- это называется сокрытие данных – один из главных принципов объектно-ориентированного программирования
- доступ к скрытым (private) данным разрешен только методам класса
- использовать ключевое слово private в объявлении класса не обязательно, поскольку это спецификатор доступа к объектам класса по умолчанию

Инициализация объекта класса

По крайней мере 3 разных способа инициализации объекта

- без начального значения

```
Hep3Vector x;
```

- с присвоением начального значения

```
Hep3Vector x(1.0, 1.0, 1.0);
```

- копирование другого объекта

```
Hep3Vector x(1.0, 1.0, 1.0);
Hep3Vector y = x;
```

- в каждом случае происходит неявный вызов специального метода класса, который называется конструктор

В этом примере – 3 конструктора класса

```
class Hep3Vector {
public:
        Hep3Vector();
        Hep3Vector(double x, double y, double z);
        Hep3Vector(const Hep3Vector &v);
        //  much more not shown
private:
        double dx, dy, dz;
};
```

Конструктор класса

реализация конструкторов

```
Hep3Vector::Hep3Vector(double x, double y, double z) {
    dx = x;
    dy = y;
    dz = z;
}
Hep3Vector::Hep3Vector(const Hep3Vector &vec) {
    dx = vec.dx;
    dy = vec.dy;
    dz = vec.dz;
}
Hep3Vector::Hep3Vector() {
}
```

- конструктор вызывается автоматически после выделения памяти при создании объекта
- когда имя класса и имя метода совпадают – эта функция конструктор
- Foo::bar() означает, что bar() является методом класса Foo (:: - операция разрешения контекста)

Методы доступа к скрытым данным

объявление класса

```
class Hep3Vector {
public:
        double x();
        double y();
        double z();
        //  much more not shown
private:
        double dx, dy, dz;
};
```

реализация методов класса для доступа к скрытым данным

```
double Hep3Vector::x() {
        return dx;
}
double Hep3Vector::y() {
        return dy;
}
double Hep3Vector::z() {
        return dz;
}
```

Реализация класса

объявление класса

```
class Hep3Vector {
public:
        double mag();
        double phi();
        double cosTheta();
        //  much more not shown
private:
        double dx, dy, dz;
};
```

файл
Hep3Vector.h

реализация остальных методов класса для операций над данными

```
#include “Hep3Vector.h”
double Hep3Vector::mag() {
        return sqrt(dx*dx + dy*dy + dz*dz);
}
double Hep3Vector::phi() {
        return dx == 0 && dy == 0 ? 0.0 : atan2(dy,dx);
}
double Hep3Vector::cosTheta() {
        double ptot = mag();
        return ptot == 0 ? 1.0 : dz/ptot;
}
```

файл
Hep3Vector.cpp

Стиль программирования

Fortran (или C)

```
common /points/hits(3,100)
real*4                 hits
real*4 x, y, z, r
! do some work ...
x = hits(1,i)
y = hits(2,i)
z = hits(3,i)
r = sqrt(x*x + y*y + z*z)
```

C++

```
Hep3Vector  hits[100];
// do some work
double r = hits[i].mag();
```

- более простой и эффективный код
- не нужно знать структуру внутренних данных
- модульность
- возможность повторного использования кода

Практическая задача (№ С6)

1. Создать класс STUDENT, содержащий следующие закрытые (private) элементы данных:

- NAME – фамилия и инициалы (char[ ])
- GROUP – номер группы (int)
- SESSION – успеваемость (массив из 5 элементов int)

и соответствующие открытые (public) методы класса для:
а) доступа к данным, т.е. ввода и вывода значений закрытых элементов данных
б) проверки наличия неудовлетворительных оценок.

2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив STUD1, являющийся массивом из 3 объектов класса STUDENT
- вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку “2”
- если таких студентов нет – напечатать соответствующее сообщение.