

Язык программирования C++

Лекция 7

Объектно-ориентированный подход:
классы

- инициализация
- замена операторов

Пример класса 'Hep3Vector'

Рассмотрим реальный класс `Hep3Vector` и некоторые дополнительные компоненты объявления класса

```
class Hep3Vector {
public:
    Hep3Vector (double x=0., double y=0., double z=0.);
    Hep3Vector (const Hep3Vector&);
    double x() const;
    double y() const;
    double z() const;
    double Phi() const;
    double cosTheta() const;
    double mag() const;
    // much more not shown
private:
    double dx, dy, dz;
};
```

- в этом примере конструктор класса использует аргументы по умолчанию
- ключевое слово `const` после функции означает запрет на изменение любых элементов объекта при вызове этой функции

Инициализация объекта класса

```
Her3Vector x(1.0, 0.0, 0.0);  
Her3Vector y = x; // C style  
Her3Vector y(x); // C++ class style
```

- обе формы инициализации объекта **y** эквивалентны
- обе вызывают копирующий конструктор
- форма с '=' допускает использование определенных пользователем преобразований типов, если типы **x** и **y** разные
- обе формы могут использоваться и для встроенных типов **int**, **float** и т.д.

Рассмотрим

```
Her3Vector x = 1.0;
```

эквивалентно

```
Her3Vector tmp(1.0);  
Her3Vector x = tmp;
```

но может быть записано и в виде

```
Her3Vector x(1.0);
```

Инициализация членов класса

Конструктор может быть реализован как и любой другой метод класса

```
Her3Vector::Her3Vector(double x, double y, double z) {  
    dx = x;  
    dy = y;  
    dz = z;  
}
```

- однако элементы данных (члены-данные) должны быть объявлены до присвоения

Альтернативная форма инициализации элементов класса

```
Her3Vector::Her3Vector(double x, double y, double z) :  
    dx(x), dy(y), dz(z) {}
```

- обратите внимание на ':' перед открывающей скобкой {
- выражение `dx(x)` напрямую вызывает конструктор
- какой из конструкторов – зависит от типа аргумента
- реализация, т.е. тело функции, необходима, даже если она будет пустой

Операторы и функции

Операторы могут рассматриваться как функции

```
double add ( double a, double b) {  
    return a + b;  
}  
double x, y, z;  
// ...  
z = x + y;  
z = add ( x, y);
```

- `add()` производит действие над двумя аргументами и возвращает результат
- оператор `+` производит действие над двумя операндами и возвращает результат

Использование математических операторов является более компактным и облегчает чтение

```
double add ( double a, double b);  
double mul ( double a, double b);  
double a, b, x, y, z;  
// ...  
z = add ( mul (a, x), mul (b, y));  
z = a * x + b * y;
```

C, C++ и Fortran – все определяют операторы для встроенных типов

Замена операторов (1)

Пример функции-оператора в объявлении класса `Hep3Vector`

```
class Hep3Vector {  
public:  
    Hep3Vector& operator += (const Hep3Vector &);  
    // more not shown  
};
```

- именем функции является ключевое слово `operator`, за которым следует символ операции
- оператор-функция вызывается в случае

```
Hep3Vector p, q;  
// ...  
q += p;
```

- функция вызывается для `q` – левой части выражения
- аргументом будет `p` – правая часть выражения
- `q += p;` - это сокращенная запись для `q.operator+=(p);`

```
Hep3Vector p, q, r;  
// ...  
r = q += p;  
// r.operator=( q.operator+=(p) )
```

Замена операторов (2)

Пример реализации функции замены оператора += :

```
Hep3Vector& Hep3Vector::operator+=(const Hep3Vector& p) {  
    dx += p.x(); // could have been dx += p.dx;  
    dy += p.y();  
    dz += p.z();  
    return *this;  
}
```

- **this** – это скрытый элемент данных класса, являющийся указателем на сам объект класса
- **this -> dx** эквивалентен **dx**
- напоминание: для указателей используем оператор доступа '->' вместо '.'

Замена операторов (3)

все операторы могут быть заменены пользователем за исключением:

`.` `.*` `::` `sizeof` `?:`

нет возможности ввести новые операторы, например `operator**()` для возведения в степень

также нельзя заменять (перегружать) операторы для встроенных типов `int`, `float` и т.д.

рекомендуется не отходить от традиционного математического смысла операторов

```
Hep3Vector p, q;  
double z;  
// ...  
z = p*q;    // ?
```

- это векторное или скалярное произведение ?
- класс `Hep3Vector` определяет его как скалярное произведение

Фортран (или C) и C++

Сумма векторов на Фортране

```
real p(3), q(3)
! ...
q(1) = q(1) + p(1)
q(2) = q(2) + p(2)
q(3) = q(3) + p(3)
```

Сумма векторов на C++, используя классы

```
Her3Vector p, q;
// ...
q += p;
```

Практическое использование классов

Программа, использующая класс
`my_programm.cpp` :

```
#include "Hep3Vector.h"
int main() {
    // ... some operators
    Hep3Vector a = Hep3Vector(1.0, 2.3, 5.4);
    // или
    Hep3Vector a(1.0, 2.3, 5.4);
    // или
    Hep3Vector *pa = new Hep3Vector(1.0, 2.3, 5.4);
    // ... some operators
    return 0;
}
```



Заголовочный файл
`Hep3Vector.h` :

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



Реализация класса
`Hep3Vector.cpp` :

```
#include "Hep3Vector.h"
Hep3Vector::Hep3Vector(double x,
    double y, double z) { ... }
double Hep3Vector::x() { ... }
...
```

Команды Linux

в общем случае:

```
g++ my_programm.cpp Hep3Vector.cpp
```

компиляция только класса:

```
g++ -c Hep3Vector.cpp
```

создается

```
Hep3Vector.o
```

компиляция с использованием откомпилированного ранее класса:

```
g++ my_program.cpp Hep3Vector.o
```

Практическая задача (№ С7)

Рассмотрим объявление класса, т. е. файл `Hep3Vector.h` :

```
class Hep3Vector {
public:
    Hep3Vector();
    Hep3Vector (double x, double y, double z);
    Hep3Vector (const Hep3Vector& v);
    double x();
    double y();
    double z();
    double Phi();
    double cosTheta();
    double mag();
private:
    double r, cos_theta, phi;
};
```

Написать минимально необходимую реализацию класса

- конструктор получает в качестве аргументов координаты `x`, `y` и `z`
- и должен инициализировать (т.е. вычислить начальные значения) скрытые элементы данных `r`, `cos_theta` и `phi`

Проверить полученную реализацию класса, используя его в программе `main()`