Объектно-ориентированное программирование

C++

Вопросы

- Каким образом применение полиморфизма позволяет упростить работу со сложными системами?
- Каким образом применение наследования позволяет упростить работу со сложными системами?
- Каким образом применение механизма инкапсуляции позволяет упростить работу со сложными системами?
- Как описать класс посредством UML?

Объектно-ориентированные языки и C++

ОО-языки программирования

1. C#
2. C++
3. Java
4. Delphi
5. Eiffel
6. Simula
7. D
8. Io
9. Objective-C
10. Object Pascal
11. VB.NET
12. Visual DataFlex
13. Perl
14. PowerBuilder
15. Python
16. Scala
17. JavaScript
18. JScript .NET
19. Ruby
20. Smalltalk
21. Ada
22. Xbase++
23. X++
24. Vala
25. PHP

Рейтинг языков программирования

Position Sep 2013	Position Sep 2012	Delta in Position	Programming Language	Ratings Sep 2013	Delta Sep 2012	Status
1	1	=	C	16.975%	-2.32%	A
2	2	=	Java	16.154%	-0.11%	A
3	4	↑	C++	8.664%	-0.48%	A
4	3	↓	Objective-C	8.561%	-1.21%	A
5	6	↑	PHP	6.430%	+0.82%	A
6	5	↓	C#	5.564%	-1.03%	A
7	7	=	(Visual) Basic	4.837%	-0.69%	A
8	8	=	Python	3.169%	-0.69%	A
9	11	↑↑	JavaScript	2.015%	+0.69%	A
10	14	↑↑↑↑	Transact-SQL	1.997%	+1.12%	A
11	15	↑↑↑↑	Visual Basic .NET	1.844%	+1.00%	A
12	9	↓↓↓	Perl	1.692%	-0.57%	A
13	10	↓↓↓	Ruby	1.382%	-0.34%	A
14	12	↓↓	Delphi/Object Pascal	0.897%	-0.10%	A-
15	16	↑	Pascal	0.888%	+0.06%	A
16	13	↓↓↓	Lisp	0.770%	-0.20%	A
17	19	↑↑	PL/SQL	0.676%	+0.07%	A-
18	24	↑↑↑↑↑↑	R	0.646%	+0.21%	B
19	20	↑	MATLAB	0.639%	+0.08%	B
20	25	↑↑↑↑↑	COBOL	0.628%	+0.20%	B

C++

- В 1980 году Бьерн Страуструп в AT&T Bell Labs стал разрабатывать расширение языка C под условным названием C++
- Первый коммерческий транслятор: 1983 год
- Главное нововведение C++ - механизм классов, дающий возможность определять и использовать новые типы данных.

Основные отличия C++ от C

1. В C++ появились классы и объекты.
2. В C++ появились две новые операции: `new` и `delete` – замещение функций `malloc` и `free`:
    1. `new` = `malloc` + конструктор,
    2. `delete` = `free` + деструктор.
3. Появился механизм обработки исключений: `try` + `catch`

Прежде чем переходить к ОО-особенностям языка C++ изучим синтактические отличия C++ от C

Встраиваемые функции

В C++ появились ключевое слово `inline`, означающее рекомендацию компилятору сделать функцию *встраиваемой*, то есть вместо генерации вызывающего её кода подставлять непосредственно её тело.

Но тело встроенной функции, подобно макросу, должно быть написано там же, где её заголовок.

Таким образом, функция

```
inline double Sqr(double x) {return x*x;}
```

будет вычисляться так же быстро, как x*x.

Аргументы по умолчанию

- В C++ Один или больше последних аргументов функции могут задаваться по умолчанию:

```
void f(int x, int y=5, int z=10);

// void g(int x=5, int y); /* Неправильно!  По умолчанию задаются только
последние аргументы */

f(1); // будет вызвано f(1, 5, 10)
f(1, 2); // будет вызвано f(1, 2, 10)
f(1, 2, 3); // будет вызвано f(1, 2, 3)
```

C и C++ - передача параметров

- В C аргументы всегда передаются по значению (происходит копирование значения аргумента в локальную функцию)
- Можно передавать указатель:

```
void foo (int *x)
{
  *x = 17;
}
int main ()
{
  int z = 5;
  foo (&z);
  /* остальные варианты больше не
имеют смысла */
  /* z теперь равно 17 */
}
```

С и С++ - передача параметров

- Передача параметров по указателю ОЧЕНЬ ОПАСНО!
- Можно забыть * в теле функции
- Или забыть & при вызове (сколько раз такое было при вызове функции “scanf”?!)
- В Си++ можно *передавать параметры по ссылкам*:

```
void foo (int& x)
{
  x = 17;
}
int main ()
{
  int z = 5;
  foo (z);
  /* z теперь равно 17 */
}
```

Ссылки в C++

- Ссылку в C++ можно понимать или как альтернативное имя объекта, или как безопасный вариант указателей. Ссылки имеют три особенности, отличающие их от указателей:
  - При объявлении ссылка **обязательно инициализируется** ссылкой на уже существующий объект данного типа.
  - Ссылка **пожизненно** указывает на один и тот же адрес.
  - При обращении к ссылке **операция * производится автоматически**.

Пространства имен

- В Си++ структуры («struct»), объединения («union») и классы («class»)стали создавать полноправные типы данных. Кроме того, каждая структура и объединение (как и, разумеется, каждый класс) получили своё полноправное пространство имён.

```
struct Foo {
  typedef unsigned char byte;
  byte data [16];
};

int main () {
  Foo::byte x;
  x = 17;
}
```

- Оператор «::» - оператор разрешения области видимости, позволяет определить по какому пути надо идти, чтобы найти необходимое имя.

Пространства имен

- Могут определяться не только в рамках классов, но и ключевым словом «namespace» :

```
mytypes.h
namespace MyFavouriteTypes {
  typedef unsigned char byte;
}
```

- С помощью директивы «using namespace» вы говорите, что «дальше я буду ссылаться на такое-то пространство имён без указания его имени». Эту директиву можно написать и внутри функции:

```
#include "mytypes.h"
int main () {
  using namespace MyFavouriteTypes;
  byte data [16];
  …
}
```

Библиотека ввода-вывода iostream

- Никаких printf и scanf!
- Каноничный Hello World на C++

```
#include <iostream>

int main()
{
    std::cout << "Hello,world!\n";
    return 0;
}
```

Библиотека iostream

- Пример ввода-вывода посредством iostream:

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

int main()
{
   int x;
   cout << "enter choice:";
   cin >> x;
   while (x < 1 || x > 4)
   {
      cout << "Invalid choice, try again:";
      cin >> x;
      // not a numeric character, probably
      // clear and pull off the non-numeric character
      if (cin.fail())
      {
         cin.clear();
         char c;
         cin >> c;
      }
   }
}
```

Используем std

Вывод

Ввод

Обработка неправильного ввода

Классы в C++

Описание классов в C++

```
class <имя> {
        [ private: ]
                [ <описание скрытых элементов> ]
        [ protected: ]
                [ <описание защищенных элементов> ]
        public:
                [ <описание доступных элементов> ]
};// Описание заканчивается точкой с запятой
```

Спецификаторы видимости private и public

- Спецификаторы доступа `private` и `public` управляют видимостью элементов класса.
- Элементы, описанные после служебного слова `private`, видимы только внутри класса. Этот вид доступа принят в классе по умолчанию.
- Интерфейс класса описывается после спецификатора `public`.
- Действие любого спецификатора распространяется до следующего спецификатора или до конца класса. Можно задавать несколько секций `private` и `public`, порядок их следования значения не имеет.
- Ключевое слово `protected` будем обсуждать более подробно при обсуждении механизма наследования

Пример класса

Конструктор (метод с именем класса, ничего не возвращает)

Встроенные (inline) методы

```
class monster
{
  int health, ammo;

public:
  monster(int he = 100, int am  = 10)
    { health = he; ammo = am;}
  void draw(int x, int y, int scale, int position);
  int get_health(){return health;}
  int get_ammo(){return ammo;}
};
```

Реализация метода класса

- Если метод определен в классе, но его реализация относительно сложна, реализацию выносят в другое место программы, чтобы не загромождать описание класса

```
void monster::draw(int x, int y, int scale,
int position)
{ /* тело метода */}
```

Использование классов в программе

- Если ранее мы определили какой-либо класс, мы можем описать его экземпляры (объекты) в нашей программе следующим образом:

```
monster Vasia;// Объект класса monster с параметрами по умолчанию
monster Super(200, 300);// Объект с явной инициализацией
monster stado[100];// Массив объектов с параметрами по умолчанию
/* Динамический объект (второй параметр задается по умолчанию) */
monster *beavis = new monster (10);
monster &butthead = Vasia;// Ссылка на объект
```

Доступ к полям и методам

- *Доступ к открытым (*public*) элементам объекта* аналогичен доступу к полям структуры:

```
объект.поле
указатель -> поле
(*указатель).поле
объект.метод( параметры )
указатель -> метод( параметры )
(*указатель).метод( параметры )
```

Итого

- C++ - ОО-язык программирования.
- Отличия C++ от C:
  - Методы описания классов и объектов
  - Inline – функции;
  - Методы обработки исключительных ситуаций
  - Работа с пространствами имен
  - Новые методы ввода-вывода
  - И многое-многое другое

Итого

- Описание классов в C++
  - Ключевое слово «`class`»
  - Модификаторы доступа `public`, `protected`, `private`
  - Конструктор – специальный метод класса, который вызывается при создании объекта.
  - Есть возможность описывать методы класса в отдельном файле, посредством операции указания области видимости «::»
  - Можно создавать объекты определенного класса (в том числе, и в динамической памяти с использованием `new`).
  - Доступ к методам и полям класса осуществляется посредством операций «.» и «->»