

Объектно-ориентированное программирование на C++

Введение

C++ — компилируемый статически типизированный язык программирования общего назначения. Поддерживая разные парадигмы программирования, сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщенного программирования.

- Разработчик — Страуструп Б., Bell Labs.
- Предшественник — C (1979), Simula-67, Smalltalk
- Дата создания — 1983 г.
- Международный стандарт - ISO/IEC 14882

Цели:

- лучше языка C;
- поддерживать абстракцию данных;
- поддерживать объектно-ориентированное программирование.

Компоненты ООП

- **Объект** - это инкапсулированная абстракция, которая включает информацию о состоянии и чётко определённое множество протоколов доступа (сообщения, которые обрабатывает объект).
- **Сообщения** - это специальный символ, идентификатор или ключевое слово, которое представляет выполняемое объектом действие.
- **Класс** - представляет определённый тип объектов и задаётся с помощью описания класса, которое определяет переменные состояния и протокол доступа к объектам данного класса. Классы организуются иерархически, причём подклассы наследуют свойства породивших их классов.
- **Экземпляр объекта** - объекты принадлежат к какому-либо классу. Свойства экземпляра объекта определяются описанием класса.
- **Метод** - метод существует для каждого сообщения, определенного для некоторого класса. Метод определяет реакцию объекта на сообщение. Объекты обрабатывают сообщения в соответствии с методами, заданными в описании класса.

Терминология в C++

- **Класс** - это новый тип данных, является расширением структурного типа данных.
- **Объект** - это переменная типа classname, где classname - определенный ранее класс.
- **Данные состояния** - закрытые данные или переменные экземпляра объекта, объявляются в описании класса и называются полями данных, данные-члены или просто члены.
- **Сообщение** - сообщения, которые объект класса может обработать указываются при помощи прототипов функций в описании класса (объявление функций).
- **Методы** в C++ - это определения функций. Прототипы функций с их определениями вместе представляют собой сообщения, которые может обработать объект. В совокупности они называются функциями-членами класса.
- **Подкласс** в C++ называют производным классом. Его родительский класс называют базовым классом.

Основные свойства объектно-ориентированного языка

- Абстракция
- Инкапсуляция
(сокрытие реализации)
- Наследование
- Полиморфизм
 - Перегрузка функций
 - Шаблоны
 - Виртуальные методы

Объектно-ориентированный подход позволяет:

- уменьшить сложность программного обеспечения;
- повысить надежность программного обеспечения;
- обеспечить возможность модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов;
- обеспечить возможность повторного использования отдельных компонентов программного обеспечения

Расширения свойств

- Размещение описаний переменных
- Обязательное использование прототипов функций
- Параметры функций по умолчанию
- Расширение области видимости переменных

```
for (int i=0; i < 5; i++)  
{  
    int k = 55; k +=i;  
    //...  
    int j = k*i;  
    //  
}
```

```
include <iostream.h>  
include "my.h"
```

```
void DrawCircle(int  
x=100, int y=50, int  
rad=20);  
//  
DrawCircle();  
DrawCircle(20);  
DrawCircle(20,5);  
DrawCircle(20,5,40);
```

```
int a = 5; int c;  
int incr( int k)  
{  
    int a = k + 15;  
    c = ::a + 4;  
}
```

- Использование перечислений

```
enum COLOR {BLACK, RED, GREEN };
COLOR c;
c = RED;
if ( c != GREEN )
{
}

enum DAYS1 { morning = 4, day = 3, evening = 2,
night = 1 };
enum DAYS2{ morning = 1, day, evening, night };
enum DAYS3{ morning, day = 2, evening, night };
```

- Использование структур как типа

```
struct Student {
    char name[80];
    int id;
    float stip;
};
//
Student st1, st2, starr[20], * pst;
```


- Присваивание структур

```
struct Student {  
    char name[80];  
    int id;  
    float stip;  
};  
//  
Student st1, st2, starr[20], * pst;  
st2 = st1;
```

- Анонимные объединения

```
static union { // глобальные - статические  
    char name[80];  
    long id;  
};  
void main(){  
    union { int number;  
            int counter;  
    };  
    counter = id;  
}
```

- new и delete

```
тип * pi = new тип  
тип * pi = new тип(значение)  
тип * pi = new тип[количество]
```

```
int * pi = new int[99];  
int * pi = (int*)malloc(99);
```

```
delete указатель  
delete [] указатель
```

- ВВОД/ВЫВОД В ПОТОКИ

```
#include <iostream.h>  
int main()  
{  
    cout << "Hello, World!";  
    int i, j;  
    cin >> i >> j;  
    cout << "i=" << i << " j=" << j << endl;  
    return 0;  
}
```

- Использование namespace

```
namespace A
{
    int i;
    int j;
};

namespace B
{
    int i;
    int j;
};

int i = B::i + A::j;
using namespace A;
i = i + 88;
j = j + 99;
```

```
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    cout << "Hello, World!";
    return 0;
}
```

```
1 #include <iostream>
2
3 class Person
4 {
5 public:
6     std::string name;
7     unsigned age;
8     void print()
9     {
10         std::cout << "Name: " << name << "\tAge: " << age << std::endl;
11     }
12 };
13 int main()
14 {
15     Person person;
16     // устанавливаем значения полей класса
17     person.name = "Tom";
18     person.age = 38;
19     // вызываем функцию класса
20     person.print();
21 }
```

/tmp/buAzjiB28t.o

Name: Tom Age: 38