# Программирование на языке C++
# Лекция 7

## Указатели на функции

Александр Смаль

# Указатели на функции

Кроме указателей на значения в C++ присутствуют три особенных типа указателей:

1. указатели на функции (унаследовано из C),
2. указатели на методы,
3. указатели на поля классов.

# Указатели на функции

Кроме указателей на значения в C++ присутствуют три особенных типа указателей:

1. указатели на функции (унаследовано из C),
2. указатели на методы,
3. указатели на поля классов.

Указатели на функции (и методы) используются для

1. параметризации алгоритмов,
2. обратных вызовов (callback),
3. подписки на события (шаблон Listener),
4. создания очередей событий/заданий.

# Указатели на функции: параметризация алгоритмов

```
void qsort (void* base, size_t num, size_t size,
          int (*compar)(void const*,void const*));
```

```
int doublecmp(void const * a, void const * b)
{
    double da = *static_cast<double const*>(a);
    double db = *static_cast<double const*>(b);
    if (da < db) return -1;
    if (da > db) return 1;
    return 0;
}

void sort(double * p, double * q)
{
    qsort(p, q - p, sizeof(double), &doublecmp);
}
```

# Указатели на функции: параметризация алгоритмов

Упростим предыдущий пример и сделаем его типобезопасным:

```
void sort(int * p, int * q, bool (*cmp)(int, int))
{
    for (int * m = p; m != q; ++m)
        for (int * r = m; r + 1 != q; ++r)
         // if ( *(r + 1) < *r )
            if ( cmp(*(r + 1), *r) )
                swap(*r, *(r + 1));
}

bool less   (int a, int b) { return a < b; }

bool greater(int a, int b) { return a > b; }

void sort(int * p, int * q, bool asc = true)
{
    sort(p, q, asc ? &less : &greater);
}
```

# О полезности `typedef`

Что здесь объявлено?

```
char * (*func(int, int))(int, int, int *, float);
```

# О полезности `typedef`

Что здесь объявлено?

```
char * (*func(int, int))(int, int, int *, float);
```

Функция двух целочисленных параметров, возвращающая указатель на функцию, которая возвращает указатель на `char` и имеет собственный список формальных параметров вида:
`(int, int, int *, float)`

# О полезности `typedef`

Что здесь объявлено?

```
char * (*func(int, int))(int, int, int *, float);
```

Функция двух целочисленных параметров, возвращающая указатель на функцию, которая возвращает указатель на `char` и имеет собственный список формальных параметров вида:
`(int, int, int *, float)`

Как стоило это написать:

```
typedef char* (*MyFunction)(int,int,int*,float);

MyFunction func(int, int);
```