Псевдонимы типов

`typedef`

`typedef` позволяет объявить псевдоним для некоторого существующего типа.

```
typedef long long ll;

long long x = 0;
ll y = 0;  // ll == long long
```

Синтаксис простой - `<имя типа> <имя псевдонима>;` и в любое место до имени псевдонима вставить слово `typedef`

```
// все объявления эквивалентны
typedef unsigned long long ull;
unsigned typedef long long ull;
unsigned long typedef long ull;
unsigned long long typedef ull;

// а так нельзя
unsigned long long ull typedef; // CE
```

typedef

Можно объявить несколько псевдонимов для одного типа в одной строке и даже для его комбинаций

```
typedef int int1_t, int2_t; // int1_t и int2_t теперь псевдонимы для int

typedef int* int_ptr_t;           // int_ptr_t == int*
typedef int int10_t[10];          // int10_t == int[10]
typedef int func_int2int(int); // func_int2int == int(int)

// то же самое, но в одну строку
typedef int int_t, *int_ptr_t, int10_t[10], func_int2int(int);
```

`typedef`

- `typedef` *не создает нового типа*, он лишь объявляет псевдоним для уже существующего

```
typedef int alias_type;

// абсолютно эквивалентные объявления
int x = 0;
alias_type x = 0;
```

- Можно объявить один и тот же псевдоним несколько раз, но он не должен менять своего значения. Менять смысл существующих типов тоже нельзя.

```
typedef int int_t;
typedef int int_t;  // Ok
typedef long int_t; // CE

typedef double float; // CE
```

`using` (C++11)

Альтернативный способ объявления - использовать `using`. Этот способ эквивалентен `typedef`, но гораздо удобнее и нагляднее.

```
using int_t = int;
using int_ptr_t = int*;
using int10a_t = int[10];
using func_int2int = int(int);
```

Синтаксис задан более жестко (`using` может располагаться только в начале) и допускается только одно определение на строку

Типы-члены класса

Псевдоним можно объявить внутри класса, тогда он станет типом-членом этого класса

```
struct S {
  typedef int int_t;

private:
  using long_t = long;
};

S::int_t x = 0;  // Ok
S::long_t y = 0; // CE
```

```
error: 'using long_t = long int' is private within this context
```

Шаблон псевдонима

Можно объявлять шаблонные псевдонимы

```
template <class T>
class Stack;

template <class T>
using StackAlias = Stack<T>;

// объявления эквивалентны
Stack<int> s;
StackAlias<int> s;

// обычно используются для сокращения длинных имен
template <class T>
using Stack3D = Stack<Stack<Stack<T>>>;

Stack3D<int> table; // Stack<Stack<Stack<int>>>
```

Шаблон псевдонима

- Шаблон псевдонима можно объявлять только с помощью `using`, `typedef` эту опцию не поддерживает

```
// не перепутай

template <class T>
using S = Stack<T>; // так можно

template <class T>
typedef Stack<T> S; // а так нельзя
```

- Шаблон псевдонима не поддерживает ни полную, ни частичную специализацию