Статические массивы, ссылки, указатели

Оператор Switch

Синтаксис:

```
switch (expression)
{
    case value_1:
        // some code
        break;
    case value_2:
    {
        // some code
    } break;

    // ...

    default:
        break;
}
```

Новый синтаксис (c++17):

```
switch (initialization; expression)
{
    case value_1:
        // some code
        break;

    // ...

    default:
        // some code
        break;
}
```

Пример использования Switch

```
void showErrorByCode(int errorCode)
{
    switch (errorCode)
    {
        case 0:
            std::cout << "Can't open file" << std::endl;
            break;
        case 1:
            std::cout << "Can't modify file" << std::endl;
            break;
        case 2:
            std::cout << "Incorrect filename" << std::endl;
            break;
        default:
            std::cout << "Incorrect error code: " << errorCode << std::endl;
            break;
    }
}
```

Комментарии

Однострочный:

```
int main() // Точка входа
{
    return 0;
}
```

Многострочный:

```
/*
    Немного комментариев коду
    не помешает
*/
```

Множественное присваивание

Операторы присваивания (=, +=, -=, …) в языке C++ являются операциями, то есть они возвращают значение, присвоенное переменной.

Это позволяет использовать конструкции вида:

```
void someFunction()
{
    int x, y, z;

    x = y = z = 0;

    x += y = 5;
}
```

Ссылочные переменные

Ссылочная переменная – служит для задания синонима другой переменной,

Обозначение:

```
тип_переменной& имя_переменой = выражение;
```

NB: Обязательно должна быть проинициализирована

Пример использования ссылочной переменной

```
void someFunction()
{
    int variable = 23;
    int& reference = variable;

    std::cout << "original: " << variable << std::endl; // 23
    std::cout << "ref: " << reference << std::endl;     // 23

    variable += 5;
    reference += 10;

    std::cout << "original: " << variable << std::endl; // 38
    std::cout << "ref: " << reference << std::endl;     // 38
}
```

Виды передачи аргументов функции

- По значению

  Функция получает копии фактических параметров

```
void increase(int value) {
    ++value;
}

int main()
{
    int x = 2;

    increase(x);

    std::cout << x << std::endl; // 2 ☹
}
```

Виды передачи аргументов функции

- По ссылке

  Функция получает сами фактические параметры

```
void increase(int& value) {
    ++value;
}

int main()
{
    int x = 2;

    increase(x);

    std::cout << x << std::endl; // 3 ☺️
}
```

Пример возврата ссылки

```
int& getValue(int& variable)
{
    return variable;
}

int main()
{
    int x = 3;
    getValue(x) = 4;
}
```

Массивы

Массив – структура данных, хранящая множество элементов одного типа

Объявление статического массива:

```
имя_типа имя_массива[количество_элементов];
```

Например:

```
int array[1000];
wchar_t array[1];
```

Массивы

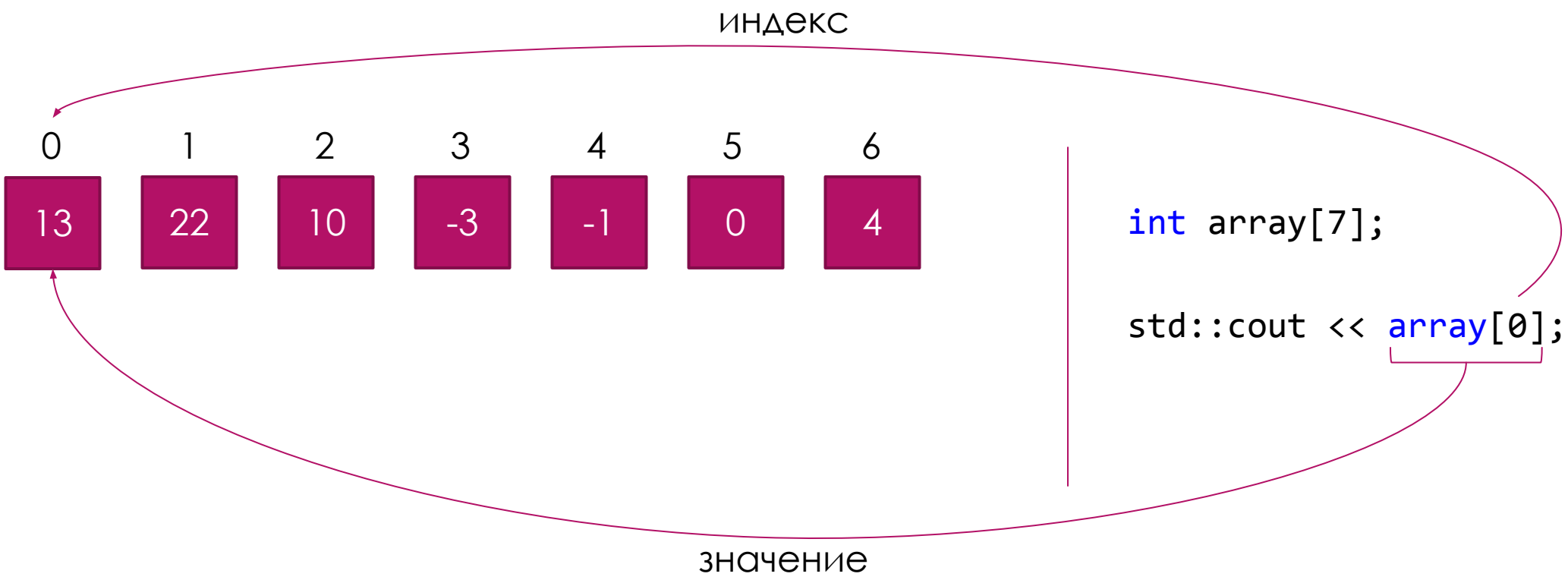
Объявление статического массива с инициализацией:

```
имя_типа имя_массива[размер_массива] = { элемент_1, элемент_2, ...
};
```

Примеры:

```
int array[3] = { 1, 2, 3 };
double array[100] = { 1, 2 };
float array[50] {};
char32_t array[] = { 1, 2 } // компилятор самостоятельно подсчитает
                // количество элементов
```

Индексация массивов



Контексты использования массива

```
array[3] = 2;
array[2] += 11;

if (array[12] == 10)
{
    // some code
}

function(array[0]);

int& fifth = array[5];
```

Размер статического массива

```
int array[];
sizeof(array); // возвращает размер массива в байтах
sizeof(array) / sizeof(array[0]); // возвращает количество элементов
```

Указатели

Указатель – переменная, значением которой является адрес другой переменной (в т.ч. и указателя), поддерживает арифметику и операцию разыменования

Объявление:

```
тип_переменной* имя_указателя;
```

Примеры:

```
int var = 3;
int* pointer = &var;
*pointer = 2;
```