

# Новые структурные операторы и массивы данных

# Оператор SWITCH (переключатель)

- Эта инструкция служит для ветвления программы во многих направлениях.
- Ее синтаксис:
- 
- **switch** (<Выражение>)
- {
- **case** <Константа 1>:
- <Последовательность инструкций 1>
- **case** <Константа 2>:
- <Последовательность инструкций 2>
- .....
- **case** <Константа N>:
- <Последовательность инструкций N>
- **default:**
- <Последовательность инструкций>
- }

# Пример

- `unsigned i;`
- `cin >> i;`
- `switch ( i )`
- `{`
- `case 0: cout << 0<<\n;`
- `case 1: cout << 1 <<\n;`
- `case 2: cout << 2 <<\n;`
- `case 3: cout << 3 <<\n;`
- `case 4: cout << 4 <<\n;`
- `case 5: cout << 5 <<\n;`
- `default: cout << "МНОГО\n ";`
- `}`

# Оператор SWITCH с BREAK

- `switch (<Выражение>)`
- `{`
- `case <Константа 1>:`
- `<Последовательность инструкций 1>`
- `break;`
- `case <Константа 2>:`
- `<Последовательность инструкций 2>`
- `break;`
- `.....`
- `case <Константа N>:`
- `<Последовательность инструкций N>`
- `break;`
- `default:`
- `<Последовательность инструкций>`
- `}`

# Пример

- `unsigned i;`
- `cin >> i;`
- `switch ( i )`
- `{`
- `case 0:`
- `cout << "ноль\n";`
- `break;`
- `case 1:`
- `cout << "один\n ";`
- `break;`
- `case 2:`
- `cout << "два\n ";`
- `break;`
- `default:`
- `cout << "много\n ";`
- `}`

# **ВАЖНО!!**

- **В качестве переключателя могут использоваться только целочисленные значения!**

# Итерационный цикл (for)

Формат записи этой инструкции:

```
for (<Инициализация>; <Условие>; <Модификация>)
```

```
{  
    <Инструкция 1>;  
    <Инструкция 2>;  
    .....  
    <Инструкция N>;  
}
```

Разделы **Инициализации**, **Условия** и **Модификации** в заголовке цикла разделяются символом **‘;’**.

# Примеры

```
int k;  
for (k = 0; k <= 9; ++k)  
    cout << k;
```

Если параметр `k` цикла используется только внутри цикла (после выхода из цикла переменная `k` больше не нужна), эту переменную можно (и лучше) определить непосредственно в разделе **Инициализации** цикла:

```
for (int k = 0; k <= 9; ++k)  
    cout << k;
```



В разделах **Инициализации** и **Модификации**  
можно управлять сразу несколькими  
параметрами цикла

```
for (int k = 1, n = 10; k <= 10; ++k, --n)  
    cout << k << " * " << n << " = " << k * n << endl;
```

Отдельные элементы разделов  
**Инициализации** и **Модификации**  
отделяются друг от друга символом ‘,’.

# Обязательно ли присутствие всех разделов?

- Любой раздел заголовка цикла может отсутствовать.
- Раздел **Инициализации**, например, может отсутствовать, когда начальные значения параметров цикла устанавливаются вне цикла, перед его началом.
- **Модификация** значений параметров цикла может осуществляться внутри тела цикла, а не в его заголовке.
- При отсутствии **Условия** продолжения выполнения цикла, цикл становится бесконечным и для выхода из него придется использовать инструкцию **break**.
- Однако, какой бы из разделов ни отсутствовал, соответствующие разделительные символы ';' в заголовке цикла должны обязательно присутствовать

# Массивы

- Массив представляет собой упорядоченную индексированную последовательность однотипных элементов с заранее определенным количеством элементов, объединенных под одним именем.
- Все массивы можно разделить на две группы: одномерные и многомерные.

# Объявление одномерного массива

Объявление в программах одномерных массивов выполняется в соответствии со следующим правилом:

**<Базовый тип элементов> <Идентификатор массива> [<Количество элементов>]**

Например:

```
int ArrInt [10], A1 [20];
```

```
double D [100];
```

```
char Chars [50];
```

```
bool B [200];
```

# Соглашение по начальному индексу

- **Значения индексов элементов массивов всегда начинается с 0.** Поэтому максимальное значение индекса элемента в массиве всегда на единицу меньше количества элементов в массиве.
- Обращение к определенному элементу массива осуществляется с помощью указания значения индекса этого элемента:

**A1 [8] = -2000;**

**cout << A1 [8]; // На экран выведено -2000**

- В этом примере, обратившись к элементу массива **A1** с индексом 8, мы, фактически, обратились к его 9-му элементу.

# Операции над элементами массива

- При обращении к конкретному элементу массива этот элемент можно рассматривать как обычную переменную, тип которой соответствует базовому типу элементов массива, и осуществлять со значением этого элемента любые операции, которые характерны для базового типа.
- Например, поскольку базовым типом массива **A1** является тип данных **int**, с любым элементом этого массива можно выполнять любые операции, которые можно выполнять над значениями типа **int**.

# Инициализация элементов массива

- При объявлении массива его можно инициализировать определенными значениями:
- **short S [5] = {1, 4, 9, 16, 25};**
- или так:
- **short S [ ] = {1, 4, 9, 16, 25};**
- 
- Во втором случае мы не указываем количество элементов массива S. Автоматически создается массив на 5 элементов в соответствии с инициализирующими значениями.
- Эти инициализации будут эквивалентны следующим операциям присваивания:
- **S[0] = 1; S[1] = 4; S[2] = 9; S[3] = 16; S[4] = 25;**
- Количество значений, указанных в фигурных скобках (инициализирующих значений) не должно превышать количества элементов в массиве (в нашем примере - 5).

# Внимание!!

- **В языке C++ не осуществляется проверка выхода за границы массивов.** То есть, вполне корректно (с точки зрения компилятора) будет обращение к элементу массива  $S$ , индекс которого равен **10**. Это может привести к возникновению весьма серьезных отрицательных последствий. Например, если выполнить присвоение  $S[10] = 1000$  будут изменены данные, находящиеся за пределами массива, а это может быть значение какой-нибудь другой переменной программы. После этого предсказать поведение программы будет невозможно. Единственный выход – быть предельно внимательным при работе с индексами элементов массивов.



# Объявление многомерных массивов

Многомерные массивы определяются аналогично одномерным массивам. Количество элементов по каждому измерению указывается отдельно в квадратных скобках:

```
int A1 [5] [3]; // Двумерный массив,  
элементами которого являются  
// значения типа int  
double D [10] [15] [3]; // Трехмерный массив,  
элементами которого являются  
// значения типа double
```

## Доступ к элементам многомерного массива

- Для доступа к определенному элементу многомерного массива необходимо указать в квадратных скобках конкретные значения всех индексов этого элемента. Например:
- 
- **cout << A1 [1] [2];** /\*На экран будет выведен элемент с индексом 1 по первому измерению и индексом 2 по второму измерению \*/

# Инициализация многомерных массивов

```
int A1 [5] [3] =  
{  
    1, 1, 1, // A[0] [0]=1; A[0] [1]=1; A[0] [2]=1;  
    2, 4, 8, // A[1] [0]=2; A[1] [1]=4; A[1] [2]=8;  
    3, 9, 27, // A[2] [0]=3; A[2] [1]=9; A[2] [2]=27;  
    4, 16, 64, // A[3] [0]=4; A[3] [1]=16; A[3] [2]=64;  
    5, 25, 125 // A[4] [0]=5; A[4] [1]=25; A[4] [2]=125;  
};
```

# Задание

- Опишите одномерный массив из 5 элементов, содержащий целые числа
- Введите с клавиатуры элементы этого массива
- Выведите элементы массива на экран
- Выведите наибольшее число в этом массиве