Программирование микроконтроллеров STM32

Лекция №2

Типы данных языка C++, основные логические операторы языка

Разработчик доц. Зубков О.В.

План лекции

- Типы данных
- Массивы и структуры
- Артфметические операции языка С
- Логические операции языка С
- Операторы сравнения
- Преобразование типов данных

Имена переменных и констант

Могут включать

- латинские буквы (A-Z, a-z)
- знак подчеркивания _
- цифры 0-9

НЕ могут включать

- русские буквы
- пробелы
- скобки, знаки +, =, !, ? и др.

Объявдение констант

Константы объявляются через ключевое слово #define

Примеры

#define my_const 10 //Целочисленная константа с именем my_const и значением 10

#define f_const 3.2 // Вещественная константа с именем my_const и значением 10

Типы данных целых чисел

- **bool** 8 bits 0 to 1
- **uint8_t** 8 bits 0 to 255
- **int8_t** 8 bits -128 to 127
- **int16_t** 16 bits -32768 to 32767
- **uint16_t** 16 bits 0 to 65535
- **int32_t** 32 bits -2^{31} to $2^{31}-1$
- **uint32_t** 32 bits 0 to $2^{32}-1$
- **int64_t** 64 bits -2^{63} to $2^{63}-1$
- **uint64_t** 64 bits 0 to $2^{64}-1$

Типы данных вещественных чисел

- **float** 32 bits ±1.18E-38 to ±3.39E+38
- **double** 64 bits ±2.23E-308 to ±1.79E+308

Описание переменных

При описании переменных указывается ее тип, имя и, если необходимо, начальное значение

Примеры

uint8_t per;

uint32_t n, k;

uint8_t m=3;

float d=2.1;

Массивы

Конечная именованная последовательность однотипных величин называется ***массивом***. Описание массива в программе отличается от описания простой переменной наличием после имени квадратных скобок, в которых задается количество элементов массива (размерность).

Элементы массива нумеруются с нуля.

Синтаксис:

тип-данных имя-массива [размер-массива]

float a [10]; **//Пример описания массива из 10 вещественных чисел**

uint8_t a[2][3]={{1,3,5},{1,2,3}}; // Двумерный массив с идентификацией начальных значений

Структуры

В отличие от массива, все элементы которого однотипны, структура может содержать элементы разных типов. В языке C++ структура является видом класса и обладает всеми его свойствами.

Синтаксис:

```
struct [имя типа]
    {
        тип_1 элемент_1;
        тип_2 элемент_2;
            …
        тип_n элемент_n;

    }   [ список_описателей ];
```

Пример структуры

```
struct {
int16_t s; /* сохранено в байтах 0 и 1 */
uint8_t c; /* сохранено в байте 2 */
uint32_t l; /* сохранено в байтах 4, 5 и 6 */
uint8_t c2; /* сохранено в байте 7 */
} s;
```

Условное представление структуры в памяти

Оператор присваивания

Общая структура:

куда записать

что

```
имя переменной = выражение;
```

Арифметическое выражение может включать

- константы (постоянные)
- имена переменных
- знаки арифметических операций:

  `+ - * / %`

- вызовы функций
- круглые скобки ( )

Примеры использования арифметических команд

```
int32_t a, b;
float  x, y;
    a = 5;   //присвоение константы 5
    b = 25; //присвоение константы 25
    x = 2.0*(0.1 + y); //вычисление зн. выражения
    a = b + (int32_t)x; //присвоение с преобразованием типа переменной x - (int32_t)x
```

При делении целых чисел остаток отбрасывается!

```
int8_t b,a = 7;
float x;
b = a / 4;
b = 4 / a;
x = float(a) / 4;
```

1

0

1.75

Сокращенная запись операций в Си

полная запись	сокращенная запись
`a = a + 1;` инкремент	`a++;`
`a = a + b;`	`a += b;`
`a = a - 1;` декремент	`a--;`
`a = a - b;`	`a -= b;`
`a = a * b;`	`a *= b;`
`a = a / b;`	`a /= b;`
`a = a % b;`	`a %= b;`

Примеры в программ

```
int32_t a, b;


a = 5;
b = a + 2;
a = (a + 2)*(b – 3);
b = a / 5;
a = a % b;
a++;
b = (a + 14) % 7;
```

Поразрядные операции

В С имеются операции, пригодные для обработки отдельных разрядов переменных. Такие операции называются поразрядными (операции с битами). Они позволяют изменять, считывать и сдвигать разряды в переменных. При этом переменная рассматривается не как число, а как комбинация двоичных разрядов, т.е. как логический код. Операция выполняется отдельно над каждым разрядом. Перечень поразрядных операций приведён в следующей таблице.

Операция	Назначение	Пример
&	*Поразрядное И*	i & 16
|	*Поразрядное ИЛИ*	i | 12
^	*Поразрядное Исключающее ИЛИ*	k ^ 10
~	*Поразрядное НЕ*	~a
<<	*Поразрядный сдвиг влево*	<< 2
>>	*Поразрядный сдвиг вправо*	>> 2

Существует краткая форма поразрядных операций присваивания. Например:

x &= y вместо x = x & y

Поразрядные операции можно использовать только с целочисленными типами данных, к вещественным числам их применять нельзя!!!

Для выполнения операции логического И используется символ & следующим образом:

```
uint8_t var = 153; //двоичная запись 10011001
uint8_t mask = 0x11; // число 00010001 (число 17)
uint8_t res = var & mask; // результат 00010001
```

В ходе выполнения двоичной операции ИЛИ результирующий бит устанавливается равным 1, если хотя бы один бит соответствующих операндов равен 1. В противном случае, результирующее значение равно 0. Для выполнения данной логической операции используется символ ‘|’ как показано ниже:

```
uint8_t var = 153; //двоичная запись 10011001
uint8_t mask = 0x11; // число 00010001
uint8_t res = var | mask; // результат 10011001
```

При операции исключающее ИЛИ результирующий бит устанавливается равным 0, если оба бита соответствующих операндов равны 1, и 1 в противном случае. Для выполнения данной операции в языке C++ используется символ ‘^’:

```
uint8_t var = 153; //двоичная запись 10011001
uint8_t mask = 0x11; // число 00010001
uint8_t res = var ^ mask; // результат 10001000
```

При выполнении операции поразрядного отрицания все биты, равные 1, устанавливаются равными 0, а все биты равные нулю, устанавливаются равными 1. Для выполнения данной операции в языке C++ используется символ ‘~’ как показано в следующем примере:

```
uint8_t var = 153; //двоичная запись 10011001
uint8_t not = ~var; //результат 01100110
```

Операция смещения битов влево определяется знаком << и смещает биты значения левого операнда на шаг, определенный правым операндом, например, в результате выполнения команды

10001010 << 2;

получится результат 00101000. Здесь каждый бит перемещается влево на две позиции, а появляющиеся новые биты устанавливаются нулевыми.

```
uint8_t var = 1;
var = var <<1; //00000010 – значение 2
var <<= 1; //00000100 – значение 4
```

Аналогично, при операции смещения вправо >> происходит сдвиг битов переменной на шаг, указанный в правом операнде. Например, сдвиг

00101011 >> 2;

приведет к результату 00001010. Здесь, также как и при сдвиге влево, новые появляющиеся биты устанавливаются равными нулю. В результате выполнения последовательностей операций

```
uint8_t var = 128; //1000000
var = var >> 1; //0100000 – значение 64
var >>= 1; //0010000 – значение 32
```

Примеры ввода значений переменных

uint8_t x;

x=(1<<0)|(1<<3); //Задается значение переменной x, у которой 0 и 3-й разряды установлены в «1», а остальные разряды равны «0».

Операции отношения

Все операции отношения используются для сравнения значений переменных или выражений. Эти операции вырабатывают значение булевского типа: ИСТИНА (true) или ЛОЖЬ (false). Численных эквивалентов для этих значений в языке C# не существует. Перечень операций приведён в таблице

Используются в сочетании с операторами проверки условия

Операция	Назначение	Пример
= =	*Равно*	I = = 0
! =	*Не равно*	K!=15
>	*Больше, чем*	Z>15.2
<	*Меньше, чем*	Tab< -132.654
>=	*Больше или равно*	Z11>=0
<=	*Меньше или равно*	Y2<=10

Логические операции

К логическим операциям относятся операция логического И (&&) и операция логического ИЛИ (||). Операнды логических операций могут быть целого типа, плавающего типа или типа указателя, при этом в каждой операции могут участвовать операнды различных типов.

Логические операции не вызывают стандартных арифметических преобразований. Они оценивают каждый операнд с точки зрения его эквивалентности нулю. Результатом логической операции является 0 или 1, тип результата int.

Операция логического И (&&) вырабатывает значение 1, если оба операнда имеют нулевые значения. Если один из операндов равен 0, то результат также равен 0. Если значение первого операнда равно 0, то второй операнд не вычисляется.

Логические операции

Операция логического ИЛИ (||) выполняет над операндами операцию включающего ИЛИ. Она вырабатывает значение 0, если оба операнда имеют значение 0, если какой-либо из операндов имеет ненулевое значение, то результат операции равен 1. Если первый операнд имеет ненулевое значение, то второй операнд не вычисляется.

Логические операторы используются для объединения нескольких условие в единое. Они применяются при применении команд проверки условий

Пример

(x<20)&&(x>4)

(y>5)||(y<0)

Операция «приведение к типу»

Эта операция используется тогда, когда необходимо преобразовать значение одного типа в значение другого типа. Это так называемое явное преобразование типов. Операция задаётся указанием имени типа в круглых скобках. Например, (byte) – преобразовать в тип байт. Рассмотрим пример:

```
int32_t n = 10;
double z;
    z = n; //Неявное преобразование
    n = (int32_t) z; //Явное преобразование
```

Понять, когда возможно неявное преобразование, а когда приведение надо делать явно, можно на основании схемы. На схеме перечислены все арифметические типы. Стрелками указаны направления, по которым автоматически осуществляется неявное преобразование (например, из byte в double). Любое обратное приведение (против стрелок) должно быть явным!!! Но надо помнить: ответственность за такое преобразование лежит на программисте. Вполне может оказаться, что при таком преобразовании будут искажены данные или потеряна точность, и система об этом не сообщит.