БАЗОВЫЕ ЭЛЕМЕНТЫ ЯЗЫКА С++. ЛИНЕЙНЫЕ ПРОГРАММЫ

3 лекция

Дисциплина «Программирование микроконтроллеров на языке С++»

Элементы языка C++

К основным элементам языка C++ можно отнести:

- символы
- переменные
- константы
- идентификаторы
- ключевые слова
- комментарии

Символы языка C++

Множество символов используемых в языке C++ можно разделить на несколько групп.

- **I группа символов** – символы, используемые для образования ключевых слов и идентификаторов. Это прописные и строчные буквы английского алфавита, а также символ подчеркивания (_); прописные и строчные буквы русского алфавита и арабские цифры.
- **II группа символов** – знаки нумерации и специальные символы (таблица 1). Эти символы используются с одной стороны для организации процесса вычислений, а с другой – для передачи компилятору определенного набора инструкций.

Допустимые символы языка C++

Символ	Наименование	Символ	Наименование
,	запятая	~	тильда
.	точка	*	звездочка
;	точка с запятой	+	плюс
:	двоеточие	-	минус
?	вопросительный знак	( )	круглые скобки
'	апостроф	{	открывающаяся фигурная скобка
!	восклицательный знак	}	закрывающаяся фигурная скобка
|	вертикальная черта	]	закрывающаяся квадратная скобка
/	дробная черта	[	открывающаяся квадратная скобка
\	обратная черта	#	номер
<	меньше	%	процент
>	больше	&	амперсанд
"	кавычки	^	логическое не
=	равно		

Символы языка C++

- **III группа символов** – управляющие и разделительные символы (пробел, символы табуляции и др.).
- **IV группа символов** – управляющие последовательности. Это специальные символьные комбинации, используемые в функциях ввода и вывода информации. Управляющая последовательность строится на основе использования обратной дробной черты (\) (обязательный первый символ) и комбинации латинских букв и цифр. (Таблица 2)

Управляющие последовательности языка C++

Управляющие последовательности	Назначение
\a	Звуковой сигнал
\b	Возврат на шаг
\t	Горизонтальная табуляция
\n	Переход на новую строку
\v	Вертикальная табуляция
\r	Возврат каретки
\f	Перевод формата

Идентификатор

Переменная представляет собой величину, значение которой во время выполнения программы может изменяться.

Константа представляют собой величину, значение которой во время выполнения программы не изменяется.

Идентификатор (имя) – последовательность цифр и букв, а также специальных символов, которая используется для символического обозначения величин (переменных и констант), функций и т.п.

- для улучшения читаемости программы следует давать объектам осмысленные имена;
- разделяйте слова, составляющие имя, знаками подчеркивания: max_length. (имя не может включать пробел)

Правила построения идентификаторов :

- для образования идентификаторов могут быть использованы строчные или прописные буквы латинского алфавита, в качестве специального символа может использоваться символ подчеркивание (_);
- первой в идентификаторе может быть только буква или специальный символ;
- одинаковые прописные и строчные буквы считаются различными символами, так как имеют различные коды (например, abc и ABC);
- компилятор допускает любое количество символов в идентификаторе, хотя значимыми являются первые 31 символ;
- идентификатор создается на этапе объявления переменной, функции, структуры и т.п. и только после этого его можно использовать в последующих операторах программы;
- идентификатор не должен совпадать с ключевыми словами, с зарезервированными словами и именами функций библиотеки компилятора языка C/C++.

Ключевые слова языка C++

Часть идентификаторов входит в фиксированный словарь ключевых слов. Эти идентификаторы образуют подмножество ключевых слов.

Ключевые слова – это зарезервированные идентификаторы, которые наделены определенным смыслом.

Их можно использовать только в соответствии со значением известным компилятору языка C/C++. Ключевые слова не могут быть использованы в качестве идентификаторов переменных, констант, функций.

Список ключевых слов языка C++

asm	auto	break	case	catch
char	class	const	continue	default
do	double	else	enum	extern
float	for	friend	goto	if
inline	int	long	new	operator
private	protected	public	Register	return
short	signed	sizeof	Static	struct
switch	template	this	Trow	Try
typedef	typeid	union	unsigned	virtual
void	volatile	while		

Программные комментарии

Хорошим стилем программирования является запись блока комментариев в начале каждого создаваемого файла с исходным текстом. Комментарии помогают человеку читать текст программы. Комментарии могут характеризовать используемый алгоритм, пояснять назначение тех или иных переменных, разъяснять непонятные места.

При компиляции комментарии выкидываются из текста программы поэтому размер получающегося исполняемого модуля не увеличивается.

Комментарии в программе выделяются зеленым цветом.

Программные комментарии

В языке C++ имеются два типа комментарий:

- **Многострочные комментарии.**

  Символы **/*** начинают комментарий, заканчивающийся символами ***/**. Вся эта последовательность символов эквивалентна символу пропуска (например, символу пробела). Это наиболее полезно для многострочных комментариев и изъятия частей программы при редактировании.

- **Однострочные комментарии.**

  Символы **//** начинают комментарий, который заканчивается в конце строки, на которой они появились. Опять вся последовательность символов эквивалентна пропуску. Этот способ наиболее полезен для коротких комментариев.

Типы переменных

Назначение любой программы заключается в обработке различных данных, поэтому программа должна уметь хранить данные, которые она использует. Данные, которые используются в программе, хранятся в оперативной памяти компьютера.

Переменная является именем участка оперативной памяти, куда можно записать информацию и откуда ее можно всегда получить. Оперативная память компьютера состоит из последовательности ячеек (байт), и каждая ячейка имеет порядковый номер, который называются адресом памяти. Для переменной может быть выделена одна или несколько ячеек, в которых и хранятся ее значения.

Типы переменных

Тип – это ключевая характеристика и объектов (переменных, констант), и функций. Тип определяет:

- какие значения может хранить объект или возвращать функция;
- какие операции можно выполнять над этими значениями;
- сколько памяти нужно выделить для хранения значения.

Типы переменных

Все типы данных в языке C++ разделяются на ***встроенные*** (фундаментальные), которые поддерживаются компилятором по умолчанию, а также ***типы, определяемые программистом.***

Существует четыре основных встроенных типов данных:

- **целый тип** - хранит числа без дробной части, например: 45,-932, 0;
- **вещественный тип** -хранит числа с дробной частью, например: 45.12, 234.542, 0.4594;
- **символьный тип** – хранит отдельные символы, например: ‘ a ‘, ‘ x ‘, ‘5’ ;
- **логический тип**, который может иметь два значения – истина или ложь.

Типы переменных

Для обозначения типа переменной используются ключевые слова, которые определяют диапазон допустимых значений и размер области памяти, выделяемой под хранение данных.

Простые базовые типы:

Целочисленные типы – char, int

- int применяется для хранения целых чисел.
- char формально также является целочисленным типом, однако обычно используется для хранения символов. Это возможно благодаря тому, что каждому символу соответствует числовой код (например, в таблице ASCII). В зависимости от реализации char может быть знаковым (–128…127) или беззнаковым (0…255).

Типы переменных

Вещественные типы – **float**, **double**.

Предназначены для хранения чисел с плавающей точкой. Точность хранения зависит от типа: float обеспечивает до 7 значащих цифр, double до 15 значащих цифр.

Логический тип – **bool**

Принимает значения true или false.

Базовые типы данных в C++

Тип данных	Размер памяти	Диапазон значений	Назначение / Особенности
char	1 байт (8 бит)	-128…127 (signed char) или 0…255 (unsigned char)	Целочисленный тип. Чаще всего используется для хранения символов (по кодам ASCII или UTF-8).
int	4 байта (32 бита)	–2 147 483 648 … 2 147 483 647	Основной тип для хранения целых чисел. Размер зависит от платформы (на 16-битных компиляторах может быть 2 байта).
float	4 байта (32 бита)	~ $\pm 3.4 \cdot 10^{38}$	Вещественный тип одинарной точности (около 6–7 значащих цифр).
double	8 байт (64 бита)	~ $\pm 1.7 \cdot 10^{308}$	Вещественный тип двойной точности (около 15–16 значащих цифр).
bool	1 байт (часто 1 бит используется логически)	true или false	Логический тип, хранит значения истины и лжи.

Литералы языка C++

В языке C++ существует четыре основных типа литералов - фиксированных значений, записанных прямо в тексте программы:

- целочисленный литерал;
- вещественный литерал;
- символьный литерал;
- строковый литерал.

Литералы языка C++

1 Целочисленный литерал.

Служит для записи целочисленных значений и является соответствующей последовательностью цифр.

Примеры целочисленных констант:

```
250                  // число 250 в десятичной системе счисления
0b11111010 // число 250 в двоичной системе счисления
0xFA              //  число 250 в шестнадцатеричной системе счисления
0372                 // число 250 в восьмеричной системе счисления
-32122291   // отрицательное целое число
0757321     // число в восьмеричной системе счисления
```

Литералы языка C++

2 Вещественный литерал. Служит для отображения вещественных значений. Он фиксирует запись соответствующего значения в обычной десятичной или научной нотациях.

- Примеры констант с плавающей точкой и константных выражений:
  - 15.752
  - 1.5752E1
  - 15752E-2
  - -0.0025
  - 25e-4
- Целая часть константы с плавающей точкой может быть опущена, например:
  - .75
  - .0075e2
  - -.125
  - -.175E-2

Литералы языка C++

3 Символьный литерал.

Символьный литерал представляет собой последовательность из одной или нескольких литер, заключенных в одинарные кавычки. Символьный литерал служит для представления литер в одном из форматов представления. Например, литера Z может быть представлена литералом 'Z', а также литералами '\132' и '\x5A'. Любая литера может быть представлена в нескольких форматах представления: обычном, восьмеричном и шестнадцатеричном.

Литералы языка C++

4 Строковый литерал.

Строковые литералы являются последовательностью литер в одном из возможных форматов представления, заключенных в двойные кавычки. Строковые литералы, расположенные последовательно, соединяются в один литерал. Строковый литерал и объединенная последовательность строковых литералов заканчиваются пустой литерой, которая используется как индикатор конца литерала.

"Hello world" //пример строкового литерала

Сравнение: Типы данных и Литералы в C++

Классификация	Что описывает	Примеры	Комментарий
Типы данных	Переменные и функции (как они хранятся в памяти и какие операции допустимы)	int, double, char, bool, string	Определяют диапазон значений, размер памяти и возможные операции
Литералы	Конкретные фиксированные значения, записанные прямо в коде	25, 3.14, 'A', "Hello"	Каждый литерал имеет свой тип по умолчанию

Инструкции языка C++

По своей сути, программы являются набором инструкций (команд), выполнение которых и приводит к решению задачи. В языке C++ имеется достаточно много различных типов инструкций.

Инструкции программы выполняются последовательно в том порядке, в котором они записаны, если нет специальных указаний о переходах. Одной из наиболее часто используемых инструкций является *инструкция-выражение*. Другой большой группой инструкций являются *инструкции управления порядком выполнения* программы.

Ставьте закрывающую фигурную скобку для каждой открывающей скобки. **Заканчивайте** все инструкции точкой с запятой.

Инструкции языка C++

В любом месте, где можно записать одну инструкцию, также можно записать составную инструкцию, которая называется ***блоком***. Блок начинается с открывающей фигурной скобки ({) и заканчивается закрывающей фигурной скобкой (}). Хотя каждая инструкция в блоке должна заканчиваться точкой с запятой, сам блок не заканчивается точкой с запятой.

Например:

```
{
    temp = a;
    a = b;
    b = temp;
}
```

Этот участок кода действует как одна инструкция и меняет местами значения в переменных a и b.

Выражения языка C++

В языке C++ все, что может вычисляться и возвращает какой-то значение является *выражением*. Любая переменная и константа возвращает значение и, следовательно, является выражением. Чаще всего выражение представляет собой последовательность переменных и констант, объединенных знаками операций. *Например, 3+2 возвращает значение 5 и является выражением.* Выражения, после которых стоит точка с запятой, являются инструкциями.

Выражения – это переменные, функции и константы объединенные знаками операций.

Переменные, функции и константы в записи выражения называются операндами. *Например, в записи y=cos(x)+x выражением является cos(x)+x, где функция cos(x) – первый операнд, переменная x – второй операнд.*

Выражения языка C++

Существует два типа выражений:

- **арифметическое выражение** (в результате вычисления получаем число; например: a + b);
- **логическое выражение**, в результате вычисления которого получается логическое значение – true (истина) или false (ложь) (например: (c < 0)).

Операции языка C++

Операции это некоторые элементарные действия, которые можно выполнить над переменными и константами.

Компилятор знает, как их выполнять над объектами встроенных типов. В языке C++ имеется большой набор операций (более 50). Примерами операций являются - сложение, умножение, присваивание и т.п. Обычно операции задаются одним символом (например: +,*,=,[ ]). Однако есть операции, которые записываются в виде вызова специальных функций.

Операции действуют над операндами, и все операнды должны быть выражениями.

Операции языка C++

Над числовыми данными определены следующие группы операций



Арифметические операции

1) Арифметические операции определены как над данными целочисленных типов, так и вещественных

+ (сумма) - (разность) * (произведение) / (целочисленное деление) % (взятие остатка) ++ (инкремент) -- (декремент)

Арифметические операции

Операции **++** и **--** применяются только к переменным и, соответственно, увеличивают или уменьшают значение переменной на 1.

Операции ++ и -- могут быть префиксными и постфиксными. Постфиксные операции выполняются после того, как значение переменной было использовано в выражении. А префиксные – до того.

Операция ++ в постфиксной и префиксной формах

1) int a=3, b=2;

```
c=a++*b++;
```

В результате получим c=6, a=4, b=3

2) int a=3, b=2;

```
c=++a*++b;
```

В результате получим c=12, a=4, b=3

Операции присваивания

Операция присваивания = – это операция, вырабатывающая значение, которое может использоваться в выражениях. Эта операция сохраняет то, что стоит с правой стороны от знака равно (=), в операнде, которая стоит с левой стороны. *Например, выражение x = a + b; присваивает значение, которое является результатом сложения a и b, переменной x.*

Эта операция не только присваивает значение переменной, но также возвращает это значение в качестве результата. Приведенная инструкция также является выражением.

Множественное присваивание – присваивание одного и того же значения нескольким переменным. *Например, a=b=c=1; или a=1; b=1; c=1;*

Операции присваивания

Операция	Обозначение	Пример	Экв.пример	Пояснение
операция присваивания-сложения	+=	var += 16	var = var + 16	Прибавляем к значению переменной `var` число 16, результат суммирования сохраняется в переменной `var`
операция присваивания-вычитания	-=	var -= 16	var = var — 16	Вычитаем из переменной `var` число 16, результат вычитания сохраняется в переменной `var`
операция присваивания-умножения	*=	var *= 16	var = var * 16	Умножаем значение переменной `var` в 16 раз, результат умножения присваивается переменной `var`
операция присваивания-деления	/=	var /= 16	var = var / 16	Делим значение переменной `var` на 16, результат деления присваивается переменной `var`
операция присваивания-остатка от деления	%=	var %= 16	var = var % 16	Находим остаток от деления и сохраняем его в переменной `var`

Операции отношения

3) Операции отношения используются для того, чтобы определить, являются ли два числа равными, или одно из них больше или меньше другого. Каждая операция отношения в результате вычисления возвращает или true(истина) или false(ложь).

Предупреждение: начинающие программировать на C++ часто путают операцию присваивания (=), с операцией равенства (==). Это может привести к серьезной ошибке в программе. **Помните,** что операции отношения возвращают значение true или false.

Операции отношения

Название	Операция	Пример	Результат
равно	==	100 == 50;	false
		50 == 50;	true
не равно	!=	100 != 50;	true
		50 != 50;	false
больше чем	>	100 > 50;	true
		50 > 50;	false
больше чем или равно	>=	100 >= 50;	true
		50 >= 50;	true
меньше чем	<	100 < 50;	false
		50 < 50;	false
меньше чем или равно	<=	100 <= 50;	false
		50 <= 50;	true

Логические операции

Часто возникает необходимость в одной записи проверить более чем одно отношение между значениями переменных. Например, "Правда ли, что x больше чем y, а y больше чем z?". В языке C++ используются три логических операции, для того чтобы выполнить такого типа проверку.

Операция	Символ	Пример
Логическое И (AND)	&&	выражение1 && выражение2
Логическое ИЛИ (OR)	||	выражение1 || выражение2
Логическое НЕ (NOT)	!	!выражение

Операция запятая

Оператор «запятая» используется для связки нескольких выражений. Левая сторона оператора «запятая» всегда вычисляется как void (то есть не выдающее значения). Это означает, что значение выражения, находящегося с правой стороны, станет значением разделенного запятыми выражения.

Например: x = (y = 3, y + 1); Сначала присваивается 3 переменной y, а затем 4. переменной x.

Скобки необходимы, поскольку оператор «запятая» имеет более низкий приоритет по сравнению с оператором присваивания.

Об операторе «запятая» следует думать как об обычном слове «и» в русском языке, когда оно используется в выражении «сделай это, и это, и это».

Объявление переменных

Для того, чтобы в программе создать переменную, ее нужно ***объявить***. Объявление это инструкция, которая вводит переменную в программу (выделяет ей память). Если переменная не объявлена, то компилятор откажется ее использовать. **Объявление переменной может находиться *в любом месте программы*, но до того, как она будет использоваться**.

При объявлении переменной необходимо указать компилятору ее имя и тип.

```
char ch;       // создаем переменную символьного типа
int count;     // создаем переменную целого типа
double pi;     // создаем переменную вещественного типа
```

В одной инструкции можно объявить более чем одну переменную одного и того же типа.

```
int year, myAge, myWeight; // целые переменные
```

Объявление переменных

При объявлении переменные (за исключением глобальных и статических) не получают никакого начального значения. Содержимое переменных после их объявления считается неопределенным.

Объявление переменной имеет следующий формат:
Тип_переменной <список переменных через запятую>;

Переменным можно задать значение, используя операцию ***присваивания*** (=). Эта операция присваивает то, что стоит с правой стороны от знака равно (=), тому, что стоит с левой стороны.

Например, можно присвоить значение переменной следующим образом:

```
int year;        //  объявляем переменную
year = 1999; //  присваиваем переменной значение
```

Объявление переменных

Эти два шага можно объединить и ***инициализировать*** значение переменной, когда она объявляется,

Например:

```
int year = 1999;
```

Инициализация – задание начального значения или списка начальных значений переменной (переменным) при объявлении.

Различие между инициализацией и присваиванием небольшое. Основная особенность инициализации заключается в том, что значение задается переменной во время ее создания.

Если в одной инструкции объявляются несколько переменных, то они все могут инициализироваться.

Например:

```
int width = 5, length = 7;
```

Категории операторов

Оператор – действие, которое должен выполнить компьютер.

Все операторы языка C++ могут быть условно разделены на следующие категории:

- оператор объявления;
- операторы ввода/вывода;
- оператор-выражение;
- операторы управления: условные операторы, к которым относятся оператор условия *if* и оператор выбора *switch*; операторы цикла (*for, while, do while*); операторы перехода (*break, continue, return, goto*); оператор возврата *return*;
- другие операторы.

Вывод

- Основные элементы языка C++ – символы, переменные, константы, идентификаторы, ключевые слова и комментарии – составляют фундамент его синтаксиса. Их правильное использование обеспечивает корректность программ, упрощает чтение кода и повышает его надёжность.
- Типы данных, операции и операторы позволяют работать с информацией различной природы, а механизм объявления и инициализации переменных задаёт основу организации вычислений.
- Таким образом, знание и понимание элементов языка C++ является необходимой базой для дальнейшего изучения его средств и разработки эффективных программных решений.