Учебный центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана

Язык программирования C

В.Г.Тетерин – Microsoft Solution Developer (Visual C++)
teterin@specialist.ru

Вячеслав Гертрудович Тетерин
Microsoft Solution Developer (Visual C++)
teterin@specialist.ru

- Вячеслав Гертрудович — ведущий преподаватель Центра, направление — программирование. Свыше 10 лет он читает курсы по основам программирования и базам данных, языку программирования C, программированию на Visual C++ в Центре.
- Он окончил кафедру инженерной кибернетики Московского института стали и сплавов (МИСиС) и более 30 лет посвятил преподаванию различных курсов программирования и применения вычислительной техники в управлении и анализе данных. 10 лет он преподавал в МИСиС на кафедре и на факультете повышения квалификации преподавателей, 10 лет обучал системных и прикладных программистов в Центральном институте повышения квалификации специалистов металлургической отрасли и,более10 последних лет он ведет курсы программирования у нас в Центре.
- Параллельно с преподаванием Вячеслав Гертрудович занимался и практическим программированием в проектах на различных языках: Algol-60, Fortran, Basic, Assembler, C/C++, Clipper и др., в том числе участвовал в проектах по разработке системы управления весовыми дозаторами (постановка задачи, алгоритм и программная реализация) (C++); разработке программного обеспечения для учебного процесса и компонентов системы «Специалист 2» (C++, C#, VB. NET); разработке программ для учета, анализа и прогноза продаж и анализа результатов деятельности для MLM-компании «Healthway» (Clarion, SQL, VBA для Excel) и др.
- Вячеслав Гертрудович — талантливый педагог, прекрасно излагает материал. Восторженные отзывы его слушателей — лучшее доказательство его высокого профессионализма. Обучил в Центре более 3000 человек.

Центр компьютерного обучения «СПЕЦИАЛИСТ» при МГТУ им. Н.Э.Баумана®

МОДУЛЬ 1

ВВЕДЕНИЕ В ЯЗЫК С

Модуль 1. Введение в язык С

- Лексемы и пробельные символы
- Основные типы данных
- Диапазоны представляемых значений
- Декларация переменных
- Константы
- Знакомство с интегрированной средой Visual C

Лексическая структура языка. Алфавит

- ***Алфавит*** языка С составляют следующие символы:
  - прописные и строчные буквы латинского алфавита: `A - Z a - z`
    - причем, прописные и строчные буквы различаются
    - это свойство называется чувствительностью к регистру символов
  - цифры: `0 - 9`
  - специальные знаки:

    `. , ; : ? ! “ ‘ + - * / % ( ) [ ] { } < = > \ & # _ ~ ^`

    - символы `@ $ ` не используются
  - пробельные символы
    - пробел,
    - символ горизонтальной табуляции - клавиша **TAB**,
    - символ вертикальной табуляции,
    - конец строки - клавиша **Enter**.

Лексическая структура языка. Лексемы и пробельные символы

- ***Лексемы*** – это наименьшие неделимые элементы языка, из которых составляются все остальные конструкции.

- Существует 6 классов лексем:

- ***1. Ключевые слова языка***
  - Ключевые слова зарезервированы в качестве служебных слов и не могут использоваться в другом смысле:

`auto`	`break`	`case`	`char`	`const`	`continue`	`default`	`do`
`double`	`else`	`enum`	`extern`	`float`	`for`	`goto`	`if`
`int`	`long`	`register`	`return`	`short`	`signed`	`sizeof`	`static`
`struct`	`switch`	`typedef`	`union`	`unsigned`	`while`	`void`	`volatile`

  - Стандарт разрешает разработчикам компиляторов и библиотек резервировать дополнительные идентификаторы для внутренних нужд.
    - Обычно, но не обязательно, такие идентификаторы начинаются с одного или двух символов подчеркивания, например:

`_asm` `_cdecl` `__LINE__` `__FILE__`

Лексемы и пробельные символы (продолжение)

- ***2. Идентификаторы***
  - Это символические имена, которыми обозначают:
    - переменные,
    - функции,
    - типы данных,
    - метки,
    - другие объекты программы, определяемые программистом.
  - Идентификатор может состоять из
    - **латинских букв, цифр и символа подчеркивания (_),**
    - *первым символом идентификатора не может быть цифра*,
    - *не рекомендуется* в качестве первого символа использовать *подчеркивание.*
  - Прописные и строчные буквы различаются, поэтому идентификаторы `index, Index и INDEX` обозначают три разных объекта.
    - Это свойство называется ***чувствительностью к регистру символов***.
  - *Значащими символами* идентификатора, согласно стандарту ANSI, являются первые **31 символ**.
    - Некоторые компиляторы, например, Microsoft Visual C++, допускают длину идентификатора до 247 символов.

Лексемы и пробельные символы (продолжение)

3. Константы
неизменяемые (постоянные) значения

вещественные	целые	символьные
•Признаки:	•Со знаком	•8-битные коды ASCII
•десятичная точка:	•10-я система:	•символ клавиатуры:
3.14159	-127	'A'
2.	127	'='
.25	+127	•ESC-последовательности:
•показатель степени:	•Без знака	'\n', '\t'
1E-10	•8-я система:	•коды в 8-й системе:
2e10	0127	'\370'
2e+10	•16-я система	•коды в 16-й системе
•общий случай:	0x7F	'\xF8'
6.2E+20	0x7f	'\xf8'

Лексемы и пробельные символы (продолжение)

- ***4. Строковые литералы***
  - Это любой символ или последовательность символов, заключённых в двойные кавычки:

```
“Hello, world!\n”
```

- ***5. Операторы (знаки операций)***
  - В языке существует 47 операторов, каждый из которых обозначается своей лексемой.
    - Например, `+` `++` `=` `==` `+=` `>` `>=` `>>` `>>=`
  - Операторы отличаются:
    - количеством операндов: `-x` `y - x`
    - приоритетом: `a = -x + y * z`
    - ассоциативностью: `x / y * z`

- ***6. Разделители и пунктуаторы***

```
* = ( ) [ ] { } , ; : … #
```

Основные типы данных

Тип данных определяет:

- объем *блока памяти*, выделяемый для хранения значений:
  - 1 байт для символьного типа
  - 4 байта для целого типа
- *структурную организацию* этого блока памяти:
  - наличие или отсутствие знакового разряда для целого типа
  - размещения знакового разряда, полей порядка и мантиссы для типа с плавающей точкой
- *диапазон* возможных значений:
  - от 0 до 255 (от 00 до FF) для символьного типа
  - от -2^{n-1} до $2^{n-1}-1$ для целого типа
- набор возможных *операторов*, применяемых к этим значениям:
  - для значений плавающего типа не определен оператор вычисления остатка от деления
  - к значениям логического типа применяются операторы отрицания, конъюнкции, дизъюнкции

Основные типы данных (продолжение)

- Простые (скалярные) типы:
  - модификаторы знака:
    - `signed`
    - `unsigned`
  - модификаторы длины:
    - `short`
    - `long`
  - символьный
    - `char` - кодировка ASCII
    - `( wchar_t` - кодировка UNICODE)
  - целый
    - `int`
  - с плавающей точкой
    - `float` - обычной точности
    - `double` - удвоенной точности
  - тип без значения
    - `void`

Основные типы данных (продолжение)

- Простые (скалярные) типы (продолжение) :
  - логический - реализован неявно
    - ноль - `false`
    - не ноль - `true`
  - указатель - для адресации памяти
- другие типы, определяемые программистом:
  - перечисления
  - массивы
  - структуры (записи)
  - объединения (смеси)
  - битовые поля

Диапазоны представляемых значений

- Целый тип

Основное описание	Эквивалентные описания	Размер в битах	Диапазон значений
`short`	`short int`, `signed short int`, `signed short`	16	от -32768 до 32767
`int`	`signed int`, `signed`	16 или 32	как `short` на 16-битных платформах, как `long` на 32-битных платформах
`long`	`long int`, `signed long int`, `signed long`	32	от -2147483648 до 2147483647
`unsigned short`	`unsigned short int`	16	от 0 до 65535
`unsigned`	`unsigned int`	16 или 32	как `unsigned short` на 16-битных платформах, как `unsigned long` на 32-битных платформах
`unsigned long`	`unsigned long int`	32	от 0 до 4294967295

Диапазоны представляемых значений (продолжение)

- Тип с плавающей точкой

Тип	Размер в битах	Диапазон значений	Разрядность
`float`	32	от 3.4E-38 до 3.4E+38	6 цифр
`double`	64	от 1.7E-308 до 1.7E+308	15 цифр
`long double`	зависит от реализации 64 или 80	от 1.7E-308 до 1.7E+308 от 1.2E-4932 до 1.2E+4932	19 цифр

- Символьный тип

Тип	Размер в битах	Диапазон значений
`char` (по умолчанию)	8	от -128 до 127
`signed char`	8	тот же
`unsigned char`	8	от 0 до 255

Декларация переменных

- Инструкция описания переменных состоит из следующих компонентов:

описатель класса памяти	квалификаторы	модификаторы	описатель основного типа	идентификатор	= инициализатор	;

```
auto          const       signed      char
extern        volatile    unsigned    wchar_t
static                    short       int
register                  long        float
                                      double
```

```
Выражение, которое
может быть вычислено
в этом месте
программы
```

- Обязательными являются
  - ***идентификатор,***
  - ***хотя бы один из предшествующих описателей,***
  - ***точка с запятой***
- При наличии квалификатора const инициализатор обязателен:

```
const int n = 10;
```

- Если несколько идентификаторов имеют одинаковый набор описателей, то их можно объединить в одной инструкции описания, причем каждый из них может иметь свой инициализатор:

```
int a = 10, b = 20, c = 0;
```

Константы

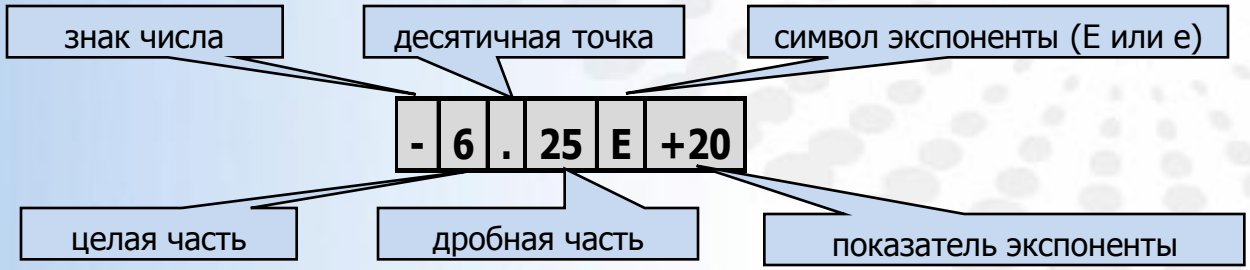
- Представление целых констант в памяти зависит от их значения:

Диапазон значений		Форма представления в памяти
		Десятичные
0	32767	целая на 16-битных платформах
32768	2147483647	длинная на 16-битных платформах
0	2147483647	целая на 32-битных платформах
2147483648	4294967295	беззнаковая длинная
		Восьмеричные
00	077777	целая на 16-битных платформах
0100000	0177777	беззнаковая целая на 16-битных платформах
01000000	017777777777	длинная на 16-битных платформах
00	017777777777	целая на 32-битных платформах
020000000000	037777777777	беззнаковая длинная
		Шестнадцатеричные
0x0	0x7FFF	целая на 16-битных платформах
0x8000	0xFFFF	беззнаковая целая на 16-битных платформах
0x10000	0x7FFFFFFF	длинная на 16-битных платформах
0x0	0x7FFFFFFF	целая на 32-битных платформах
0x80000000	0xFFFFFFFF	беззнаковая длинная

- Если следом за константой стоит суффикс `L` или `l`, то она будет представлена в памяти как длинная, например, `123L`
- Если за константой следует суффикс `U` или `u`, то она представляется как беззнаковая
- Оба суффикса могут присутствовать одновременно

Константы (продолжение)

- Вещественные константы
  - константа с плавающей точкой состоит из следующих компонентов:



- Либо целая, либо дробная часть могут отсутствовать, но не обе сразу
- Десятичная точка или **E** с порядком могут быть опущены, но не одновременно
- Константа может быть положительной и отрицательной

  - В С любая вещественная константа считается значением с двойной точностью - `double`
    - Можно определить вещественную константу и с одинарной точностью - `float`, для этого необходимо записать константу с суффиксом **F** или `f`, например, `1.25F`

Константы (продолжение)

- Символьные константы
  - Символьная константа - это некоторый символ из набора символов кода ASCII,
    - заключенный в одиночные кавычки, например, `'A'`
    - или его восьмеричное `'\101'`
    - или шестнадцатеричное `'\x41'` представление
    - Символы, имеющие специальное назначение, могут быть представлены с помощью так называемых управляющих последовательностей (**Esc** - последовательностей)

Последовательность	Значение	Обозначение в ASCII	Примечание
\0	0x00	NUL	Нуль-символ, пусто
\a	0x07	BEL	Звонок
\b	0x08	BS	Шаг назад
\f	0x0C	FF	Перевод формата
\n	0x0A	LF	Новая строка
\r	0x0D	CR	Перевод каретки
\t	0x09	HT	Горизонтальная табуляция
\v	0x0B	VT	Вертикальная табуляция
\\	0x5C	\	Обратный слеш
\'	0x2C	'	Апостроф
\"	0x22	"	Кавычки
\?	0x0F	?	Вопросительный знак

Константы (продолжение)

- Строковые константы

  – Строковая константа изображается последовательностью символов кода ASCII, заключенной в кавычки, и может содержать внутри себя Esc-последовательности, например:

```
"This is a string\n"
```

  - Компилятор представляет строки как массивы символов, в конце каждой строки он добавляет нулевой символ `'\0'`, отмечающий конец данной строки

  - Символ `\` и следующий за ним символ новой строки игнорируются

  - Это позволяет переносить продолжение длинной строковой константы на новую строку программы

```
"This is a long\
character string"
```

  - Разрешается записывать длинные строковые константы и иным способом

```
"This is a long"
"character string"
```

  - В этом случае производится сцепление (конкатенация) нескольких строковых констант в одну

Константы (продолжение)

- Перечислимые константы
  - Символические имена, указанные в описании перечисления (нумератора)

```
enum bool { no, yes };
```

  трактуются как целые константы (этот тип будет рассмотрен позже)

- Именованные константы
  - С помощью директивы препроцесора **#define** можно определить символические имена для констант любого типа
    - эти имена по традиции записываются прописными буквами

```
#define     MAXBUF   81
#define     ESC      0x1B
#define     PI       3.14159
#define     MSG      "Press any key to continue\n"
```

    - В процессе обработки текста программы *препроцессор* заменяет каждое вхождение символического имени соответствующей константой
    - Определения такого типа часто используют для введения констант - параметров реализации

Практическая работа

- Вот классическая программа “Hello, world!”:

```
#include <stdio.h>
int main()
{
    printf(“Hello, world!\n”);
    return 0;
}
```

- 1. Проведите лексический анализ программного кода
- 2. Наберите код программы в среде разработки и выполните ее компиляцию, построение и запуск

Структура простой программы

- Простая программа на С размещается в единственном *исходном модуле* (текстовом файле с расширением .c) и имеет следующую структуру:

  – комментарий, описывающий назначение программы

  – директивы **`#include`** для библиотечных функций

  – функция **`main`**,

  в теле которой располагаются:

  - описания локальных переменных
  - исполняемые инструкции
    - ввод (с подсказкой)
    - вычисления
    - вывод
  - оператор завершения (необязательный)

`prog.c`

```
/* эта простая программа на С
   вычисляет сумму двух целых чисел
*/
#include <stdio.h>

int main()
{
    int a, b, c;
    printf("Enter 2 int values: ");
    scanf("%d %d", &a,&b);
    c = a + b;
    printf("%d+%d=%d\n",a,b,c);
    return 0;
}
```

Итоги

- В этом модуле Вы изучили:
  - Алфавит и лексику языка C
  - Встроенные скалярные типы данных и их внутреннее представление
  - Синтаксис инструкции описания переменных
  - Способы записи констант и определение их типа
  - Классическую программу “Hello, world!”
  - Общую структуру простой программы, состоящей из 3-х типовых фаз:
    - Ввод исходных данных
    - Обработка
    - Представление результатов

Вопросы?

- В.Г.Тетерин – Microsoft Solution Developer (Visual C++)
- teterin@specialist.ru

ПРИЛОЖЕНИЕ

ЗАДАЧИ

Задачи

1. Вычислить среднее арифметическое двух значений x1 и x2.
2. Перевести длину, заданную в дюймах, в сантиметры (10 дюймов = 254 мм).
3. Перевести длину, заданную в сантиметрах, в дюймы.
4. Перевести температуру из шкалы Фаренгейта в шкалу Цельсия (формула для пересчета c°=(5/9)(f°-32)).
5. Перевести температуру из шкалы Цельсия в шкалу Фаренгейта.
6. Вычислить площадь кольца (внешний радиус R, внутренний - r).

Задачи

7. Вычислить периметр и площадь:
   1. квадрата со стороной, равной a.
   2. прямоугольного треугольника с катетами a и b.
   3. равнобедренного треугольника с основанием a и высотой h.
   4. равнобокой трапеции с длинами оснований a и b и высотой h.
   5. равностороннего треугольника со стороной, равной a.
8. Вычислить площадь поверхности и объем:
   1. цилиндра (радиус r, высота h).
   2. полого цилиндра (радиусы R и r, высота h).
   3. шара ($S=4\pi r^2$, $V=(3/4)\ \pi r^3$).