Программирование на языке C

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

Немного истории (начало)

- **1969–1972** — на базе языка B («би») — упрощенного варианта BCPL (Basic Combined Programming Language — Мартин Ричардс, Кембриджский университет, 1966) — начинается проектирование нового языка программирования
- **1972** — сотрудник Bell Telephone Laboratories Деннис Ритчи создает язык C («си») для разработки ОС UNIX на платформе PDP-7
- **1973** — на язык C перенесен значительный фрагмент ядра Unix для PDP-11, ранее разработанный на языке ассемблера
- **1978** — в США выходит книга Брайана Кернигана и Денниса Ритчи с описанием языка C, надолго ставшая неформальным стандартом для программистов (K&R C)

  Kernighan, Brian W.; Ritchie, Dennis M. *The C Programming Language*. Englewood Cliffs, NJ: Prentice Hall (1978)

Немного истории (продолжение)

- **1985** — книга Б. Кернигана и Д. Ритчи переведена на русский язык

  Керниган Б., Ритчи Д., Фьюер А. Язык программирования Си. Задачи по языку Си / Пер. с англ. — М.: Финансы и статистика, 1985. — 279 с.
- **1988** — выходит в свет 2-е издание книги Б. Кернигана и Д. Ритчи (первое описание будущего стандарта ANSI C)

  Kernighan, Brian W.; Ritchie, Dennis M. The C Programming Language (2nd ed.). Englewood Cliffs, NJ: Prentice Hall (1988)
- **1989** — созданный в 1983 г. комитет Американского института стандартов (ANSI) ратифицирует стандарт X3.159-1989 "Programming Language C" (ANSI C, Standard C, C89)
- **1990** — Международная организация по стандартизации (ISO) принимает стандарт ANSI C как стандарт ISO/IEC 9899:1990 (C90)

Немного истории (окончание)

- **1992** —2-е издание книги Б. Кернигана и Д. Ритчи выходит в России

  Керниган Б., Ритчи Д. Язык программирования Си / Пер. с англ. — М.: Финансы и статистика, 1992. — 272 с.

- **1999** — опубликован стандарт ISO/IEC 9899:1999 (C99)

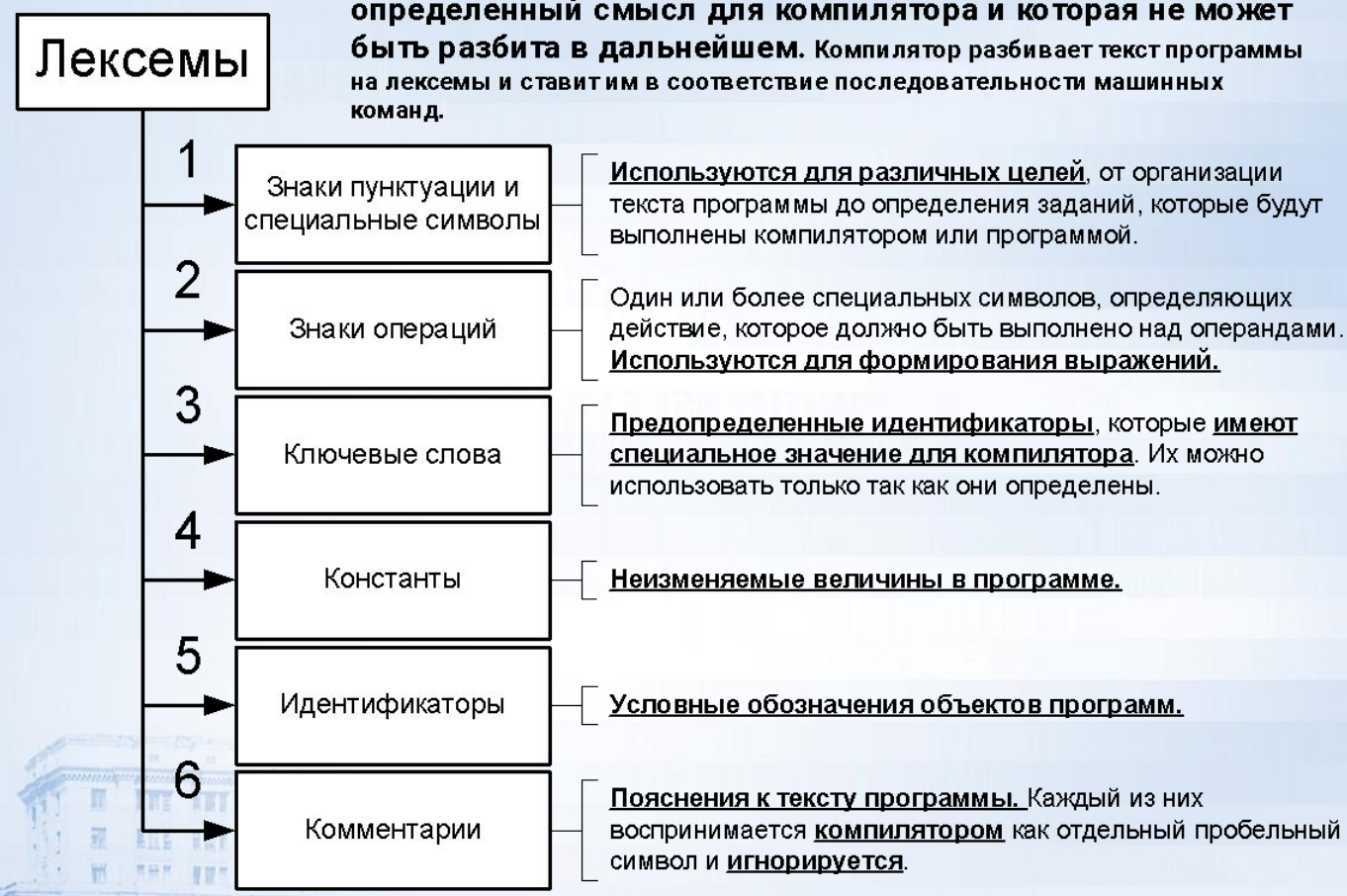
  к числу стандартных возможностей языка добавлены встроенные (inline) функции, новые типы данных (long long int, complex и др.), массивы переменной длины, макроопределения с переменной местностью и однострочные комментарии

- **2007** — начата работа над новым вариантом стандарта (так называемым “C1x”)

Алфавит и лексемы языка С

- **Алфавит**
  - буквы: A, B, C, …, Z, a, b, c, …, z
  - цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - специальные символы: + − / % . ? ! " < > | \ ' _ & ~ ^
  - знаки пунктуации языка: [ ] ( ) { } , ; : … * = #
  - пробельные символы: ␣ (пробел), ↳ (символ табуляции), ↲ (символ перевода строки)
  - прочие символы — только в комментариях к тексту программы
- **Лексемы** — идентификаторы, ключевые слова, константы, операции, разделители
  - единицы текста программы, которые при компиляции воспринимаются как единое целое и по смыслу не могут быть разделены на более мелкие элементы [Под04]

Лексема - это единица текста программы, которая имеет определенный смысл для компилятора и которая не может быть разбита в дальнейшем. Компилятор разбивает текст программы на лексемы и ставит им в соответствие последовательности машинных команд.



Границы лексем определяются:

1) пробельными символами
2) другими лексемами

Идентификаторы и ключевые слова

- **Идентификатор** — любая последовательность букв A, B, C, …, Z, a, b, c, …, z, цифр 0, 1, …, 9 и символов подчеркивания _ , не начинающаяся с цифры и имеющая длину не более 31 символа. Строчные и прописные буквы в идентификаторах различаются компиляторами
- **Ключевое слово** — одно из слов языка, входящих в следующий список:
  - спецификаторы типов: `char`, `double`, `enum`, `float`, `int`, `long`, `short`, `struct`, `signed`, `union`, `unsigned`, `void`, `typedef`
  - квалификаторы типов: `const`, `volatile`
  - квалификаторы классов памяти: `auto`, `extern`, `register`, `static`
  - операторы языка и идентификаторы специального назначения: `break`, `continue`, `do`, `for`, `goto`, `if`, `return`, `switch`, `while`; `default`, `case`, `else`, `sizeof`
  - модификаторы и псевдопеременные: конкретный набор зависит от компилятора

Константные значения (начало)

- **Константа** — неизменяемое арифметическое значение целого, вещественного, символьного или перечислимого типа, нулевой указатель либо строковый литерал:
  - целые — записываются в системах счисления по основаниям 10, 8, 16:
    - (целочисленный) нуль в любой системе счисления — `0`
    - десятичные — последовательность десятичных цифр, не начинающаяся с нуля
    - восьмеричные — последовательность восьмеричных цифр, начинающаяся с нуля
    - шестнадцатеричные — последовательность шестнадцатеричных цифр, начинающаяся с 0x или 0X
  - вещественные — записываются в десятичной системе в следующих форматах:
    - [+|-]<*целая часть*>.[<*дробная часть*>]
    - [+|-]<*целая часть*>{e|E}[+|-]<*порядок*>
    - .[<*дробная часть*>][{e|E}[+|-]<*порядок*>]

    где<*целая часть*> есть целая часть абсолютной величины десятичной мантиссы,
    <*дробная часть*> — дробная часть абсолютной величины десятичной мантиссы,
    <*порядок*> — абсолютная величина десятичного порядка (экспоненциальной части числа)

Константные значения (окончание)

- символьные — записываются естественным образом* или посредством ESC-последовательностей**, *** согласно следующим правилам:
  - * символы, имеющие экранное представление — любой входящий или не входящий в алфавит языка единичный символ в обрамлении апострофов (');
  - ** *ряд* символов, лишенных экранного представления — одна из следующих управляющих последовательностей: `'\n'` — перевод строки; `'\t'` — горизонтальная табуляция; `'\r'` — возврат каретки; `'\\'` — обратная косая черта; `'\''` — апостроф; `'\"'` — двойная кавычка; `'\0'` — нулевой символ; `'\a'` — звонок; `'\b'` — возврат на одну позицию; `'\f'` — перевод страницы; `'\v'` — вертикальная табуляция; `'\?'` — знак вопроса;
  - *** любой символ — собственный восьмеричный код в виде '\ooo', где o — цифра от 0 до 7, либо шестнадцатеричный код в виде '\xhh' или '\Xhh', где h — цифра от 0 до F;
- перечислимые — задаются в определении программистом собственного типа-перечисления;
- нулевой указатель — единственная неарифметическая константа, представимая различными компиляторами как `0`, `0L` или `NULL` (значение NULL может не совпадать с нулем (`0`) и (или) нулевым символом (`'\0'`));
- строковый литерал — заключенная в двойные кавычки (") последовательность символов, записанных по правилам для символьных констант *, **, *** без обрамляющих апострофов

Знаки и приоритет операций (начало)

Приоритет операций	Знаки операций	Порядок выполнения операций с равным приоритетом
1	( ) [ ] -> .	слева направо
2	! ~ + − ++ −− & * (<*имя типа*>) sizeof	справа налево
3	* / %	слева направо
4	+ −	слева направо
5	<< >>	слева направо
6	< <= >= >	слева направо
7	== !=	слева направо
8	&	слева направо
9	^	слева направо

Знаки и приоритет операций (окончание)

Приоритет операций	Знаки операций	Порядок выполнения операций с равным приоритетом
10	|	слева направо
11	&&	слева направо
12	||	слева направо
13	?:	справа налево
14	= *= /= %= += -= &= ^= |= <<= >>=	справа налево
15	,	слева направо

Комментарии – пояснения к тексту программы.

Например,

```
/*Это комментарий языка С*/
/*
Может
быть
многострочным /*, но не может быть вложенным*/
*/
//Это однострочный комментарий языка   С++ и стандарта С99
```

Разделители

- **Разделитель** — парный или одиночный знак пунктуации, входящий в следующий список:

```
[ ]    ( )    { }    ,    ;    :    …    *    =    #
```

Пример – Привет, Мир!

Операции в рабочем пространстве

1. Создание проекта (консольного приложения)

1.1. Меню:**File·New ⇒ Диалог "New", Вкладка "Projects".**
1.2. Выбрать из списка название проекта "Win32 Console Application".
1.3. Поле ввода "Project Name": Ввести с клавиатуры имя проекта, например, first.
1.4. Поле ввода "Location": Обратить внимание на то, где локализована папка проекта в файловой системе (нажав кнопку  можно изменить размещение этой папки).
1.5. Нажать кнопку **OK ⇒ Окно мастера "Win32 Console Application - Step 1 of 1".**
1.6. Установить переключатель в положение "An empty project".
1.7. Нажать кнопку "Finish" ⇒ Диалог "New project information". Просмотреть информацию о созданном проекте.
1.8. Нажать кнопку OK. Новый проект создан!!!

2. Создание нового файла (исходного модуля) в проекте

2.1. Меню:**File·New ⇒ Диалог "New", Вкладка "Files".**
2.2. Выбрать из списка тип файла "C++ Source File"[1].
2.3. Поле ввода "File Name": Ввести с клавиатуры имя файла (для файла на языке С обязательно явно указать расширение .c!!!), например, hello.c.
2.4. Флажок "Add to project": должен быть установлен!!!
2.5. Нажать кнопку OK. Новый файл (исходный модуль) в проекте создан!!!

Компиляция, редактирование связей и выполнение программы

Эти операции могут быть выполнены 3 способами каждая:

1 сп. С помощью командной кнопки панели **Build MiniBar** ;

2 сп. Соответствующей командой меню **Build;**

3 сп. Нажатием соответствующего сочетания клавиш.

Компиляция:	Кнопка	**Build·Compile;**	**Ctrl+F7 .**
Редактирование связей (компоновка):	Кнопка	**Build·Build;**	**F7 .**
Выполнение:	Кнопка	**Build·Execute;**	**Ctrl+F5 .**

Переменные и константы

- **Описание переменных**

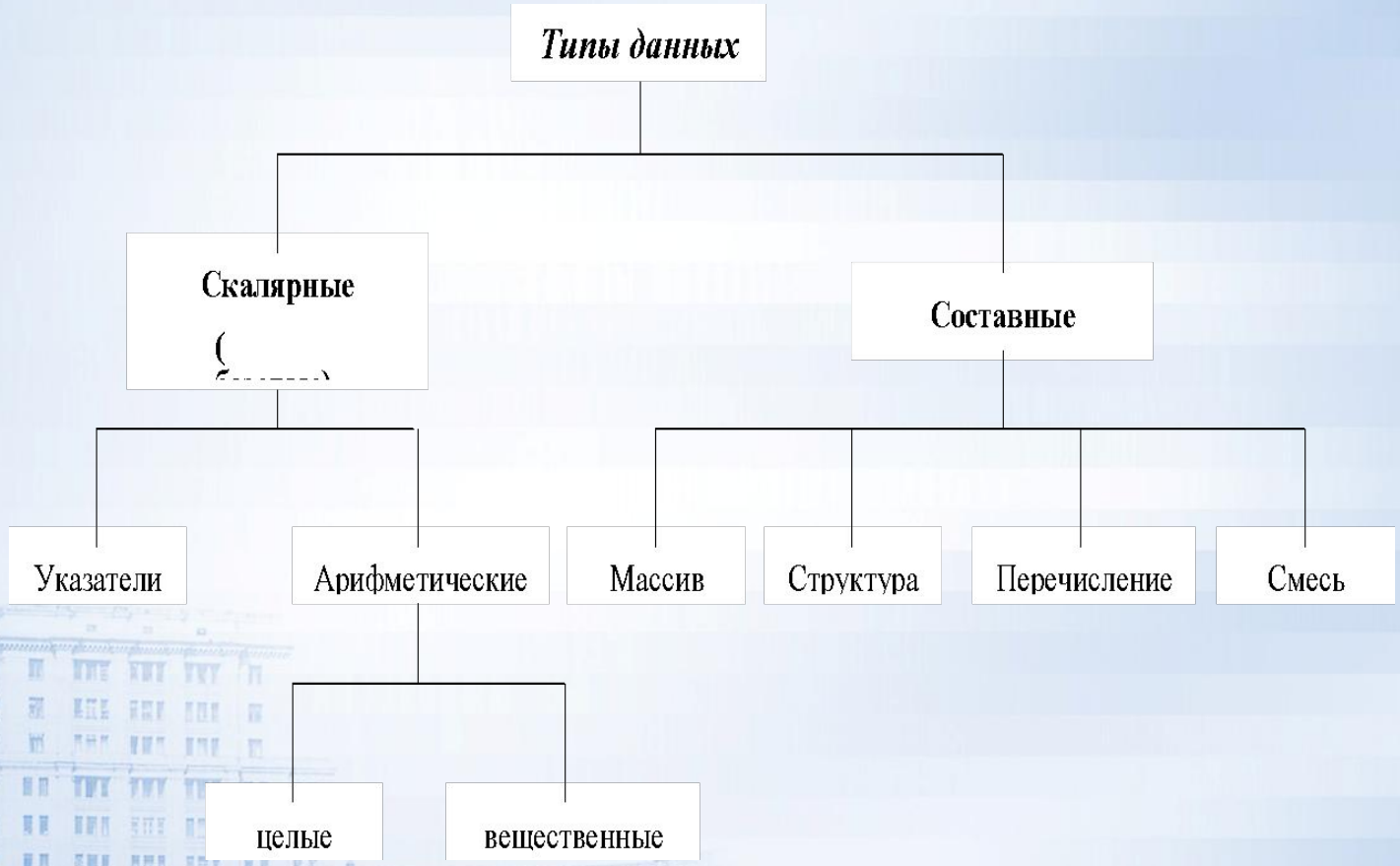
  <имя типа> <переменная 1>[[= *<значение 1>*], …, *<переменная N>* [= *<значение N>*]];

- **Описание констант**

  `const` [*<имя типа>*] *<имя константы>* = *<значение константы>*;

  - при опускании типа константы подразумевается int

  `#define` *<имя константы>* = *<значение константы>*


Типы данных
Скалярные
Составные
Указатели
Арифметические
Массив
Структура
Перечисление
Смесь
целые
вещественные

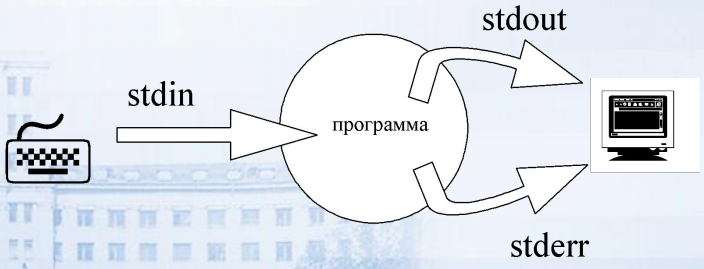
Основные типы данных

Имя типа	Размер области памяти (бит)	Диапазон значений (для вещественных типов — по абсолютной величине)
unsigned char	8	0 ... 255, '\x00' ... '\xFF'
[signed] char	8	-128 ... 127
unsigned short [int]	16	0 ... 65535
[signed] short [int]	16	-32768 ... 32767
enum	32	-2147483648 ... 2147483647
unsigned [int]	32	0 ... 4294967295
[signed] int	32	-2147483648 ... 2147483647
unsigned long	32	0 ... 4294967295
[signed] long	32	-2147483648 ... 2147483647
float	32	3.4E-38 ... 3.4E38
[long] double	64	1.7E-308 ... 1.7E308

Форматный ввод-вывод

В языке программирования С **нет встроенных средств ввода-вывода**.
В С **все операции ввода-вывода данных ассоциируются с понятием «поток»**.
«Поток» – абстракция, которая находится между программой и конкретным устройством.
Понятие потока позволяет программисту не заботится о взаимодействии программы с конкретными устройствами в составе вычислительной системы.
При запуске программы автоматически открываются и подключаются к ней **5 стандартных потоков (3** основных и 2 вспомогательных).

Основные стандартные потоки:	***Вспомогательные потоки:***
stdout –выходной	stdprn –принтера (параллельного порта)
stdin –входной	stdaux – последовательного порта
stderr –сообщений об ошибках	



Функция printf - обеспечивает форматный вывод данных в поток stdout.

Формат функции:

printf(<*форматная (управляющая)строка*> [, <*выражение 1*>,...,<*выражение N*>]);

Где форматная строка может включать в себя следующие компоненты:

- допустимые в строковой константе символы алфавита языка – выводятся на экран без изменения;
- специальные символы;
- спецификаторы форматов (начинаются с символа %) – определяют порядок вывода и представление выводимых данных.

Символ типа	Тип аргумента	Формат вывода
d,i,u,	int	Десятичное целое со знаком
o	int	Восьмеричное целое без знака
x	int	Шестнадцатеричное целое без знака, использующее "abcdef"
X	int	Шестнадцатеричное целое без знака, использующее "ABCDEF"
f	double	Значение со знаком в формате [-]dddd.dddd, где dddd - одна или более десятичных цифр
e	double	Значение со знаком в формате IEEE. Символ "e" строчный (маленький)
E	double	Значение со знаком в формате IEEE. Символ "E" прописной (большой)
g	double	Значение печатается в формате "f" или "e" в зависимости от того, какой из них более компактен для данного значения и точности
G	double	Идентичен формату "g" за исключением того, что экспоненту представляет символ "E"
c	int	Одиночный символ
s	строка	Символы строки печатаются до первого нулевого символа ('\0') или до достижения значения заданной точности
p	Указатель far на void	Печатается адрес, указанный аргументом, в формате xxxx:yyyy, где xxxx - сегмент, yyyy - смещение, а x и y - строчные шестнадцатеричные цифры

Функция scanf - обеспечивает форматный ввод данных из потока stdin.

Формат функции:

scanf(<*форматная (управляющая)строка*> [,<*адрес переменной 1*>,...,<*адрес переменной N*>]);

Где форматная строка может включать в себя следующие компоненты:

- допустимые в строковой константе символы алфавита языка – должны быть повторены при вводе;
- специальные символы - должны быть повторены при вводе;
- спецификаторы форматов (начинаются с символа %) – определяют порядок ввода и преобразование вводимых данных.

Назначение **префиксов в спецификации формата вывода**

Символ префикса	Назначение
h, l	Префиксы, которые **определяют размер ожидаемого аргумента:** **h - используется как префикс перед типами d, i, o, x, X - для указания того, что тип аргумента short int, или с u - для указания того, что тип аргумента short unsigned int;** **l - используется как префикс перед типами d, i, o, x, X - для указания того, что тип аргумента long int, или с u - для указания того, что тип аргумента long unsigned int; он также используется с типами e, E, f, q, G - для указания того, что тип аргумента double, а не float** (%lf, %le,%lg – для **ввода double, но для вывода только long double).**

Список литературы

- [КР92] Керниган Б., Ритчи Д. Язык программирования Си / Пер. с англ. — М.: Финансы и статистика, 1992. — 272 с.
- [КР06] Керниган Б., Ритчи Д. Язык программирования С / Пер. с англ. — М.: Вильямс, 2006. — 304 с.
- [Под04] Подбельский В.В., Фомин С.С. Программирование на языке Си. – 2-е доп. изд. – М., Финансы и статистика, 2004. – 600 с.