Языки программирования

Преподаватель:

Ядрышников Олег Дмитриевич

Старший преподаватель кафедры Автоматики

Языки программирования

Лекции 1 раз в 2 недели

Лабораторные работы 1 раз в 2 недели, начиная с 3 (4) недели

РГР сдать до начала сессии

Экзамен устный

Язык С (Си)

Литература

1. Б.Керниган, Д.Ритчи. Язык программирования Си.
2. М.Уэйт, С.Прата, Д.Мартин. Язык Си. Руководство для начинающих.
3. Н.Джехани. Программирование на языке Си.
4. М.Болски. Язык программирования Си. Справочник.
5. …

Методички

1. Методические указания к лабораторным работам… - № 3709
2. Методически указания к курсовым работам (Си и Ассемблер) - № 3181

Язык С (Си)

- Си (англ. C) — компилируемый статически типизированный язык программирования общего назначения.
- Разработан Деннисом Ритчи в 1972 г.в лаборатории Bell Labs (США).
- Изначально предназначен для написания ОС UNIX для ЭВМ семейства PDP-11
- Прообраз – язык B (Би)

Особенности языка С

Структурированность: Программа состоит из законченных конструкций (операторов). Позволяет избежать ошибок при программировании.

- Это свойственно для всех современных языков программирования

Эффективность: Компилятор Си генерирует компактный объектный код. Многие операции языка Си однозначно соответствуют командам Ассемблера.

- Это скорее достоинство компилятора
- Компактность (оптимальность) кода во многом зависит от программиста
- Современные компиляторы – оптимизирующие

Особенности языка С

Переносимость (мобильность): простота переноса программы на Си на ЭВМ другого семейства (с другой системой команд).

- Возможность переноса определяется наличием компилятора. Все программы на языках высокого уровня – переносимы
- Перенос выполняется на уровне объектных библиотек.
- Практически все библиотечные функции Си написаны на Си

Особенности языка С

«Невысокий» уровень: Включает средства для работы на машинном уровне. Некоторые конструкции транслируются в 1-2 машинные команды.

- Первоначально Си предназначен для PDP-11, поэтому некоторые операции позаимствованы из ASM PDP-11

Недостатки языка C

- **Сложный и не всегда удобный синтаксис**
- **Сложность определения порядка выполнения операций**
- **Несколько версий языка**
- **Плохая диагностика при компиляции и выполнении**

Область применения С

- **системное программирование**
- **системы управления базами данных (СУБД)**
- **управляющие программы (альтернатива ASM)**

C++ - объектно-ориентированное программирование

C# - разработка приложений для платформы Microsoft.NET

Java - приложения для встроенных систем, апплеты, серверное ПО

Язык С. Пример программы

```
#include <stdio.h>
int main (int argc, char* argv[])
{
   double x, y, z;
   printf (“\nВведи два числа:”);
   scanf (“%f %f”, &x, &y);
// %f - ввод/вывод чисел типа float и double;
   z = x + y;
   printf (“\n%7.3f + %7.3f = %10.4f”, x, y, z);
// %7.3f   - вывод числа float в поле 7 позиций, 3 знака после
запятой;
}
```

Язык С. Синтаксис

Программа.

Состоит из функций – автономных программных модулей.
Все функции равноправны. Выполнение начинается с main().

```
main()
{
    …
}

func1(…)
{
   …
}
```

Язык С. Синтаксис

Идентификаторы (имена)

- Первый символ – буква или «_»
- Остальные – буквы, цифры, знак «_»
- Длина произвольная (до 32 символов различаются)
- Верхний и нижний регистры – различаются
- Пробелы недопустимы

Примеры:

- *Alfa alfa*
- *Array_of_Data*
- *My_variable_with_big_name*
- *x*
- *a*

Язык С. Синтаксис

Операторы (ключевые слова)

- Зарезервированы
- Пробелы внутри недопустимы (кроме *go to*)
- Только нижний регистр
- В одной строке допустимо несколько операторов

Примеры:

```
if (a > 0) x = 3; else x = z+5;
while (a[i] < 0) n++;
for (I = 0; I < 10; i++)  s +=a [i];
```

Язык С. Синтаксис

Простой оператор: заканчивается “**;**”

a = b+c;

if (a > 0) x = 3;

Пустой оператор: *;*

- используется в зависимости от синтаксиса

Составной оператор (блок): *{ операторы }*

- может быть вложенным
- после *“}“* не ставится *“;”* (есть исключения)

Язык С. Синтаксис

Пример использования вложенных составных операторов:

```
k=0;
for (i=0; i <100; i++)
{
        if (a[i] == 0)
        {
                k++;  n = 3;
        }
}
```

Язык С. Синтаксис

Комментарии (2 типа)

- вложенность комментариев одного типа запрещена

Многострочный

```
/* ……..
   ……..
   ……..
*/
```

- Используется для больших фрагментов текста или комментирования частей программы на время отладки
- Можно использовать внутри оператора

Однострочный

```
// …….
// …….
// …….
```

- Используется для комментирования отдельных строк в программе

Язык С. Синтаксис

Пример использования комментариев в программе

```
/* функция вычисляет среднеквадратическое значение
элементов вещественного массива произвольной длины
*/
      int func (a, b)
      double *a;  // указатель на массив данных
      int b;          // размер массива
      {
         for(…)     // цикл вычисления суммы элементов
         {
             …..
         }
      /* ---- этот кусок пока не отлажен
          if (b > 1000) {       }
      */
      }
```

Язык С. Препроцессор

- **Обрабатывает текст программы перед компиляцией**
- **Директивы препроцессора начинаются с “#”**
- **Синтаксис отличается от синтаксиса С**

#include <файл>
#include “файл”

В текст программы включить содержимое указанного файла. Допускается до 10 уровней вложенности

Примеры:
#include <stdio.h> *- файл в системном каталоге*
#include “myfile.h” *- файл в текущем каталоге*
#include “c:\lib\mylib.c” *- файл в указанном каталоге*

Язык С. Препроцессор

#define идентификатор подстановка

Задание (подстановка) констант

Примеры задания констант:

#define PI 3.141595

#define LEN 100

*#define ppp LEN * 2 + PI*

x = PI / 2; □ *x = 3.141595 / 2;*

while (i < LEN) c[i++] = 0; □ *while (i < 100) c[i++] = 0;*

Язык С. Препроцессор

#define идентификатор (параметры) подстановка

Задание (подстановка) макросов

Пример задания макроса: $fun(x,y) = x^2 + y^2$

*#define fun(x, y) x * x + y * y*

Использование:

z = fun (2, 3); → *z = 2 * 2 + 3 * 3;*

z = fun (a+2, b-4); → *z = a + 2 * a + 2 + b – 4 * b - 4;* *(???)*

Корректный вариант макроса:

*#define fun(x, y) ((x) * (x) + (y) * (y))*

Язык С. Препроцессор

#define

Примеры нестандартного использования

```
#define BEGIN {
#define END }
#define если if
#define FOR for
…
#define DEBUG   // метка DEBUG определена, но значение не определено
#undef DEBUG    // отмена определения
```

Язык С. Препроцессор

#if #else #endif

Директивы условной компиляции
Допускается вложенность

Полный вариант

```
#if выражение
// если выражение ≠ 0
  Операторы 1
#else
// если выражение  = 0
   Операторы 2
#endif
```

Сокращенный вариант

```
#if выражение
// если выражение ≠ 0
   Операторы 1
#endif
```

Выражение – константное (включает только константы)

Язык С. Препроцессор

#if #else #endif

Пример

```
#define DEBUG 1        // 1 – отладка, 0 – рабочий вариант
#if DEBUG
  printf (“\nОтладочный вывод»);
#endif
…..
#if DEBUG
  x = 100;
#else
  x = 1000;
#endif
```

Язык С. Препроцессор

#ifdef #ifndef

Проверка существования константы
Значение константы не проверяется

Пример

```
#define DEBUG
    …..
#ifdef DEBUG
  printf("\nОтладочный вариант");
  x = 100;
#else
  x = 1000;
#endif
```

Язык С. Препроцессор

#pragma

Задание режимов работы компилятора
Специфичны для разных версий компиляторов

```
#pragma pack(1)  // упаковка элементов данных с точностью
// до байта. По умолчанию – до слова
        // (четный  адрес)

#pragma check_stack (off) // отключение контроля стека
        // по умолчанию включен
```

Язык С. Типы данных

Описание	Диапазон значений	Размер (байт)
char	-128 … + 127	1
unsigned char	0 … 255	1
short (short int)	-32768 … +32767	2
unsigned short	0 … 65535	2
long (long int)	-2147483648 … +2147483647	4
unsigned long	0 … 4294967295	4
int	short / long	2 / 4
unsigned int	unsigned short / unsigned long	2 / 4
fload	$\pm 10^{\pm 38}$	4
double	$\pm 10^{\pm 76}$	8
long double	$\pm 10^{\pm 76}$	10

Язык С. Типы данных

sizeof()

Вычисляет размер типа данных или переменной (массива)
По смыслу - константа

Пример использования

```
int a;
n = sizeof(int);
```

или

```
n = sizeof(a);
```

Язык С. Константы

- **Целые десятичные**: *123, +12, -456, 5*
- **Вещественные**: *12.3, -3.5, +1.2, 0.35, -.273, 5.*
- **Вещественные в экспоненциальной форме**:
  1.253e-5, -.12e4, 25E+04
  1.253e-5 □□ *1.253*10^{-5}*
- **Длинные целые (long):** *1234567l, 1234567L, 0L*
- **Шестнадцатеричные:** *0x1BC0, 0X33FE*
  (префикс *0x* или *0X*)
- **Восьмеричные:** *025, 037, 07773* (префикс 0)
- **Символьные:** *‘A’, ‘c’, ‘n’*
  (unsigned char, ASCII-код символа)

Язык С. Константы

- **Специальные символьные константы**

\n <CR><LF> переход к новой строке

\r <CR> переход к началу строки

\f <FF> перевод формата (начало новой страницы)
или очистка экрана

\t <TAB> табуляция

\b <BACKSPACE> возврат на шаг

\a <BELL> «звонок» (звуковой сигнал)

\’ апостроф

**** backslash

\27 символ с ASCII-кодом 27 (непечатный)

Язык С. Константы

- **Строковые константы**

“это строка текста” → массив *char*

Последний элемент массива – двоичный 0 (автоматически)

Пустая строка: “” содержит один нулевой байт

Перенос длинных строк:

Вариант 1:

“начало длинной строки ” // нет “;”или “,”

“продолжение длинной строки”

Вариант 2:

“начало длинной строки \ // нет кавычки

продолжение длинной строки” // с начала строки

Язык С. Константы

Правила определения типов констант

1. Десятичные константы –
   int или *long* (в зависимости от значения)

2. Восьмеричные или шестнадцатеричные константы –
   unsigned int или *unsigned long*

3. Вещественные константы –
   double

4. Явное указание типа константы имеет приоритет:

12L - *long*

13.5F - *float*

Язык С. Классы памяти

Определяют способы выделения памяти под переменные

auto	память выделяется при входе в блок и освобождается при выходе из блока. Размещается в стеке. Начальное значение произвольное.
static	память отводится при трансляции, значение сохраняется. Начальное значение 0.
register	как *auto*, но используются регистры процессора. Только для переменных типа *int*.
extern	описаны в другом файле. Память не выделяется (выделена в другом файле).

Язык С. Классы памяти

Дополнительные классы памяти в С++

const	не модифицируемый тип данных (константа) используется как альтернатива *#define*
volatile	Значение переменной меняется вне программы например, по прерываниям от таймера. Компилятор не оптимизирует обращение к этой переменной.

Пример:

```
volatile int a;

b = a;    // при каждом присваивании
c = a;    // могут быть
d = a;    // разные значения
```

Язык С. Описания переменных

Все переменные должны быть описаны.

Структура описания: [класс памяти] [тип] имя [, имя] ;

Умолчания: класс памяти: *auto* (внутри блока)
static (вне функций)
тип: *int*

Примеры описаний:

unsigned long b;
static double x, y, z;
extern unsigned char d;

static m; □□ *static int m;*
double z; □□ *auto double z;* (или *static double z;*)
p; □□ *static int p;* (или *auto int p;*)

Язык С. Описания переменных

Место описания переменных:

a) **Вне функций.**
   - класс памяти – *static* (по умолчанию) или *extern*

b) **Внутри составного оператора (блока)**
   - в начале блока
   - класс памяти – *auto* (по умолчанию), *register, static*

c) **Внутри оператора (не всегда корректно)**

```
for (int a = 0; a < 100; a++)
{
    …..
}
```

Язык С. Описания переменных

Инициализация (задание начальных значений) переменных

Для переменных *static* – при компиляции
Для переменных *auto* – при выполнении (присваивание)

Примеры:

```
int a=3, b=2;
double x, y=3.5, z;
char ch1 = 'A', ch2 = 'b'

for(i = 0; i < 20; i++)
{
   static int a = 5;   // один раз при компиляции
    int b = 2;    // при каждом проходе цикла
    ….
    a = 4;
    b = 3;
}
```

Язык С. Описания переменных

Область действия (видимости) переменных

- от места описания до конца блока (файла)

файл **main.c**

```
int a;
static int b;

main()
{
    int c;
    for (…)
    {
        int d;
        ….
    }
}
```

перегрузка данных

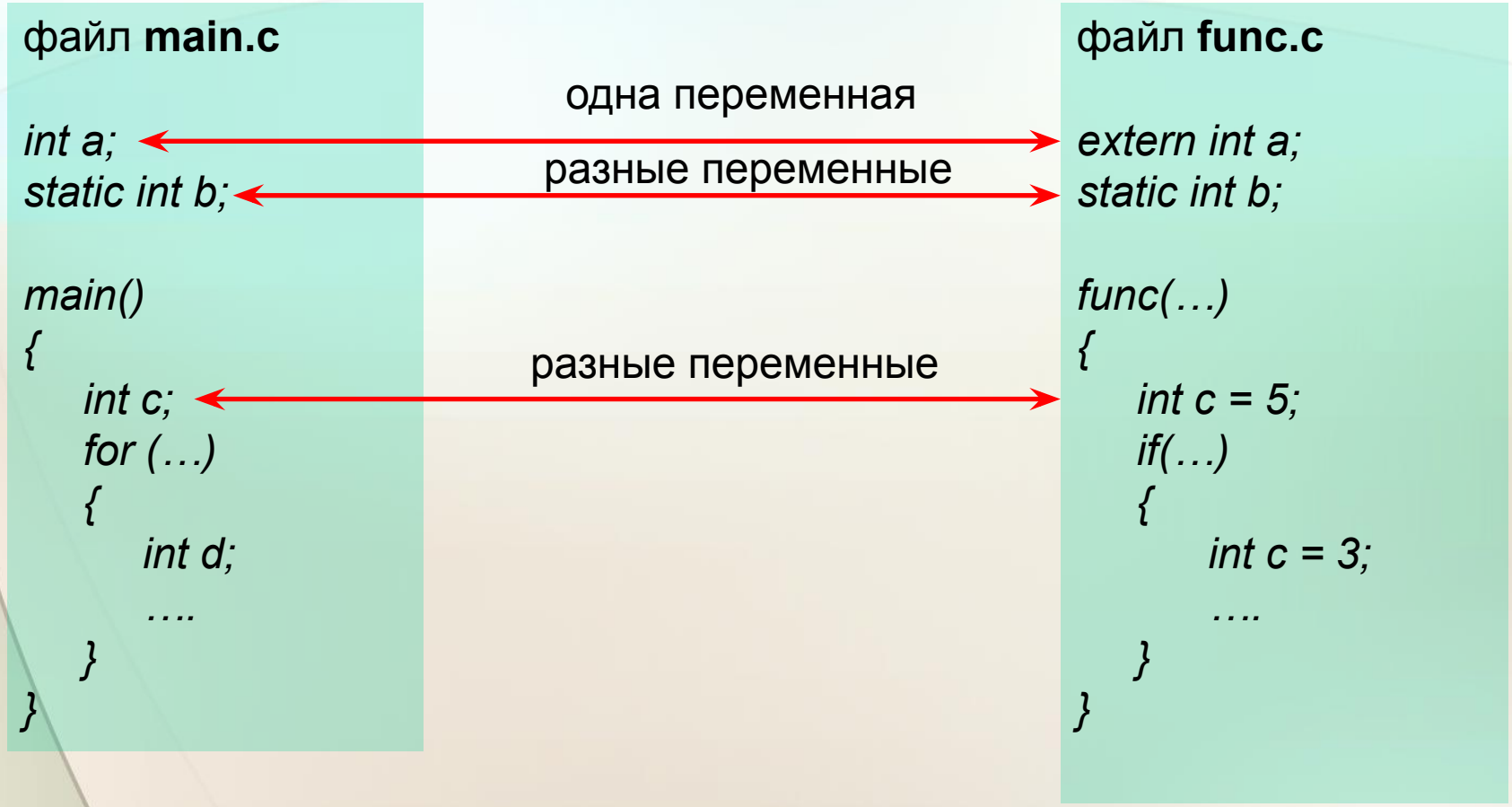
файл **func.c**

```
extern int a;
static int b;

func(…)
{
    int c = 5;
    if(…)
    {
        int c = 3;
        ….
    }
}
```

Язык С. Описания переменных

Область действия (видимости) переменных

- от места описания до конца блока (файла)



Язык С. Описания переменных

typedef

Оператор переопределения типов:

- задает синоним существующего типа данных
- улучшает «читабельность» программы
- упрощает написание машинно-независимых программ

typedef <существующий тип> <синоним>;

Пример:

typedef static unsigned int word;

word x; □□ *static unsigned int x;*

Язык С. Массивы. Описание

Массив: набор элементов данных одного типа, расположенных в памяти последовательно.

- количество размерностей произвольное
- каждый индекс в отдельных скобках
- значение индекса начинается с 0
- максимальный размер определяется средой выполнения

Примеры описаний:

int x[5]; // 5 элементов, индекс изменяется от 0 до 4
double z[20]; // 20 элементов, индекс 0…19

*int a[3][2]; // 3*2 = 6 элементов (3 строки по 2 элемента)*

a[0][0], a[0][1], a[1][0], a[1][1], a[2][0], a[2][1]

char s[100][100][100];

Язык С. Массивы. Инициализация

Одномерные массивы:

```
int a[5] = { 1,2,5,8,13}; // после “}” ставится “;”

int b[5] = { 2,3,,6};       // b[0] = 2, b[1] = 3, b[3] = 6

int c[5] = { ,,3 };         // c[2] = 3
```

Многомерные массивы:

```
int a[3][2] = { 1,2,3,4,5,6 };  // a[0][0], a[0][1], a[1][0], a[1][1], a[2][0], a[2][1]

int b[3][2] = {
            { 1, 2 }, // a[0][0], a[0][1]
            { , 4 },   // a[1][1]
            { 5 }      // a[2][0]
        };
```

Язык С. Символьные массивы.

Инициализация символьного массива (строки):

char str[20] = { ‘t’, ‘e’, ‘x’, ‘t’, ‘\0’ }; □□ *char str[20] = “text”;*

char str[ ] = “big text”; // размерность определяет компилятор
8+1=9

char str [ ] ; // массив размерности 0 == простая переменная

Многомерные массивы строк

```
char str[3][50] = {
        “строка 1”,
        “вторая строка”,
        “еще одна строка”
        };
```

Язык С. Выражения

Элементы выражений:

- константы
- переменные
- элементы массивов
- знаки операций
- вызовы функций
- скобки

Примеры:

a) *x * 23 + fun(a, b) / 12.5*
b) *x + (a - 2) * (b[12] + (b[11] / 4 - z) / 124.*
c) *x*
d) *12*

Язык С. Операции

Стандартные операции

- бинарные (2 операнда):

```
a + b      // сложение
a - b      // вычитание
a * b      // умножение
a / b      // деление

a % b      // деление по модулю (остаток от деления a на b)
           // только для целых типов данных
20 % 3 =  2
```

- унарные (1 операнд):

```
-a         // унарный минус (смена знака)
```

Язык С. Операции

Специальные операции (заимствованы из ASM)

++ инкремент (увеличение на 1)
-- декремент (уменьшение на 1)

a++; (или *++a;*) □□ *a = a + 1;*

Может использоваться самостоятельно или в составе выражений

```
   a = 3;
1) x = a++;    // a = 4, x = 3 (инкремент после присваивания)
2) x = ++a;    // a = 4, x = 4 (инкремент перед присваиванием)

a = 3; b = 5;
x = a+++++b;□□  x = (a++) + (++b);
                x =   3   +   6
                a = 4
                b = 6
```

Язык С. Оператор присваивания

<переменная> = <выражение>;
<элемент массива> = <выражение>;

Примеры:

a = b+3;
c[5] = x + func(y, x+2) / 2.5;
d = d+5;

z = (a=3) + (b=2); *a = 3; b = 2; z = a + b;*

a = b = c = d = 0; *// порядок действий – справа налево*

a += 3; *a = a + 3;*
*b *= c + 4;* *b = b * (c + 4);*

Язык С. Преобразование типов

1. Определяется порядок действий в соответствии с приоритетами
2. В каждой из операций выполняется приведение типов данных:

   - *char* → *int;* *float* → *double;* - безусловно
   - *int* → *long* → *double* - по приоритету

3. Тип результата безусловно преобразуется к типу переменной в левой части оператора присваивания.

При использовании в одной операции данных *signed* и *unsigned* правильный результат не гарантируется!

```
int x = -1;            // двоичное представление
unsigned int y = 65535;    // одно и то же
```

$x = y = 1111\ 1111\ 1111\ 1111_2$

Язык С. Преобразование типов

Пример 1

int a; double b;

1) *a = 2 / 3;*	*int / int*	□ *int*	□ *int*	*a = 0*
2) *b = 2 / 3;*	*int / int*	□ *int*	□ *double*	*b = 0.0*
3) *a = 2 / 3.;*	*int / double*	□ *double*	□ *int*	*a = 0*
4) *b = 2 / 3.;*	*int / double*	□ *double*	□ *double*	*b = 0.66666*

Пример 2

double a; int b = 2, c = 3;

a = b / c; □ *a = 0.0*

a = (double) b / c; □ *a = 0.66666*

явное преобразование типа данных

Язык С. Битовые операции

Применяются только для целых типов данных
(*char, short, long, int*)

& - “И”
| - “ИЛИ”
^ - “исключающее ИЛИ”
>> - сдвиг вправо
<< - сдвиг влево
~ - инверсия (унарная операция)

Возможные варианты записи:

&= |= ^= >>= <<=

Операции выполняются поразрядно

Язык С. Битовые операции

Таблица истинности

A	B	&	|	^
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Примеры:

a = 0010 1011
b = 1011 1010

a & *b* = 0010 1010
a | *b* = 1011 1011

Язык С. Битовые операции

Использование битовых операций (маскирование)

1) Установка разрядов

```
a = b | 4;      // установка 3 разряда (маска 00000100₂)
```

2) Сброс разрядов

```
a = b & ~4;    // сброс 3 разряда (маска 11111011₂)
```

3) Выделение разрядов

```
a = b & 4;      // сброс всех разрядов, кроме 3
```

4) Поразрядное сравнение

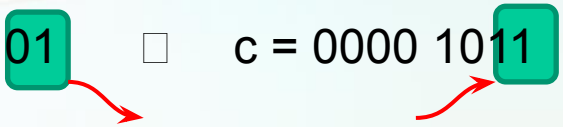
```
a = b ^ c;      // a – единицы в несовпадающих
                // разрядах b и c
```

Язык С. Битовые операции

Операции сдвига

1) *c = a >> 2;*

a = 0010 1101 □ c = 0000 1011



Сдвиг вправо на 1 разряд = деление на 2
Сдвиг влево на 1 разряд = умножение на 2

Сдвиг вправо работает по-разному для *signed* и *unsigned*

Пример:

unsigned char a = 1 ; a >> 1 = 0
char b = 1; b >> 1 = 0
char c = -1; c >> 1 = -1 (!)

Для отрицательных чисел – расширение знака

Язык С. Таблица операций

Приоритет	Операции	Порядок
1	( ) [ ] -> .	□
2	! ~ ++ -- - *(тип)* * & *sizeof()* (унарные)	□
3	* / %	□
4	+ -	□
5	<< >>	□
6	< <= >= >	□
7	== !=	□
8	&	□
9	^	□
10	|	□
11	&&	□
12	||	□
13	?:	□
14	= += -= *= /= и т.п. (все присваивания)	□
15	,	□

Язык С. Учет приоритетов

Пример 1

*x = a * 2 + b * 4;*
1 3 2

x = a << 1 + b << 2;
2 1 3

x = (a<<1) + (b<<2);
1 3 2

Пример 2

a) *if (a = b + c > 0) x = 3;*
3 1 2

b) *if ((a = b + c) > 0) x = 3;*
2 1 3

Язык С. Операции отношения

Операнды – выражения любых типов
Результат – целое *число int*: 1 – истина, 0 – ложь
Могут использоваться в арифметических выражениях

>	больше
>=	больше либо равно
<=	меньше либо равно
<	меньше
==	равно
!=	не равно

Примеры:

```
a = b >3;   // если b > 3, a = 1; иначе a = 0;

x =  a > 0 – a < 0;    // a < 0  x = -1
                       // a = 0  x = 0
                       // a > 0  x = 1
```

Язык С. Логические операции

Логические операции
Объединяют операции отношения

Операнды – целые числа (≠0 – истина, =0 – ложь)
Результат - целое число *int*: (1 – истина, 0 – ложь)

&& И
|| ИЛИ
! НЕ (унарная операция)

Примеры:

a < b < c □ *a < b && b < c*
!(a > b) □□ *a <= b*
!a // =1, если a = 0; =0, если a ≠ 0

Язык С. Условный оператор

if - else

Синтаксис:

1) *if (выражение) оператор1; else оператор2;*
2) *if (выражение) оператор1; // краткая форма*

Если *выражение* ≠ 0, выполняется *оператор1*; иначе – *оператор2*

Оператор один, но может быть составным.
Допустима вложенность

Выражение – как правило, операция отношения

Язык С. Условный оператор

Примеры:

1) *if(a > 0) x = 3;*
2) *if(a == 3) x = 5; else x = 7;*

3) *if(a) x = 4;* □□ *if(a != 0) x = 4;*
4) *if(!a) x = 4;* □□ *if(a == 0) x = 4;*

5) *if (a == 5) z = 0;*
6) *if (a = 5) z = 0;* □ *if((a = 5) != 0) z = 0;* □ *z = 0;*

7) *if (a > 0) { x = 1; y = 2; c = 3; }*
8) *if (a == 'c' || a == 'C') x = 0;*

Язык С. Условный оператор

Вложенные условия:

```
if ( x > 0 )
{
    c = 3;        // x > 0
    y = x + 5;
}
else       // x <= 0
{
    if ( a < 3 )
    {
        c = 5;    // (x <= 0) && (a < 3)
        z = y – x;
    }
}
```

```
if ( x > 0 ) { c = 3; y = x + 5; } else { if (a < 3) { c = 5; z = y - x; } }
```

Язык С. Условная операция

Условная операция – тернарная (3 операнда)

<выражение1> ? <выражение2> : <выражение3>

Если *<выражение1>* ≠ 0, значение = *<выражение2>*
иначе значение = *<выражение3>*

Допустима вложенность
Порядок действий – справа налево.

x = a > 0? b + 3 : c - 5; *if (a > 0) x = b + 3; else x = c - 5;*

x = (a > 0)? 1 : -1;

x = a > 0? 1 : a < 0? -1 : 0; *// x = sign (a);*
вложенность

Язык С. Переключатель

switch	case default break

```
 switch (<выражение>)
{
   case <конст1>: операторы1;   // <выражение> = <конст1>
   case <конст2>: операторы2;   // <выражение> = <конст2>
            break;              // прекратить выполнение
   case <конст3>: операторы3;   // <выражение> = <конст3>
   default: операторы 4;        // нет совпадений
}
```

<выражение> - целое (*int, char, long*)
<конст> - константы целого типа
<операторы> - любой набор операторов или пустая строка

default – может отсутствовать; располагается в любом месте внутри ***switch***

Язык С. Переключатель

Примеры:

1)
```
switch (n)
{
  case 1:  a = sin(x);   // если n = 1
       break;
  case 2:  a = cos(x);  // если n = 2
       break;
  default:  printf (“Error”);   // в остальных случаях
}
```

2)
```
char ch;
switch (ch)
{
   case ‘s’:  a = sin(x);   break;
   case ‘c’:  a = cos(x);  break;
   default:  printf (“Error”);
}
```

Язык С. Переключатель

Примеры:

3)
```
char ch;
switch (ch = getchar())
{
   case ‘s’:
   case ‘S’:  a = sin(x);   break;

   case ‘c’:
   case ‘C’:  a = cos(x);  break;

default:  printf (“%c - Error”, ch);
}
```

Язык С. Циклы. Цикл for

for

Синтаксис:

for *(<оператор1>; <выражение>; <оператор2>) <оператор3>;*

<оператор1> - инициализация; 1 раз перед началом цикла
<оператор2> - модификация; каждый раз после тела цикла
<оператор3> - тело цикла (м.б. составным)
<выражение> - условие продолжения цикла (≠ 0)

- Любой из элементов может отсутствовать, “**;**” - обязательны
- Если отсутствует <выражение> - бесконечный цикл
- *<оператор1>* и *<оператор2>* - м.б. несколько операторов через “**,**”
- Условие продолжения проверяется перед телом цикла (цикл с предусловием)
- Допускается вложенность

Язык С. Циклы. Цикл for

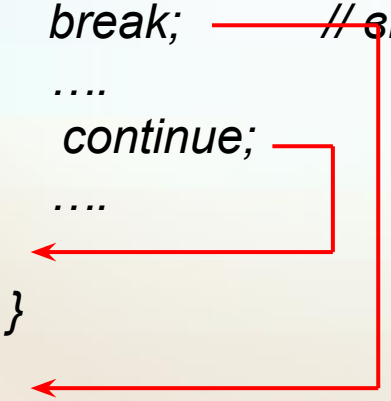
Примеры:

```
1) for ( i = 0; i < 10; i++)  s = s + a[ i ];          // i = 0 … 9

2) for (i = 0, s = 0; i < 10; i++ )  s += a[ i ];

3) for ( i = s = 0; i < 10; s += a[ i++ ] )  ;   // пустое тело цикла

4) for ( ; ; ) { … }                    // бесконечный цикл

5) for ( i = 0; i < 100; i++)                  // вложенные циклы
    {
        for ( j = 0; j < 100; j++)
        {
      s += a[ i ][ j ];
      if (a[ i ][ j ] != 0) n++;
        }
     }
```

Язык С. Циклы. Цикл for

break; continue;

```
for ( i = 0; i < 100; i++)
{
   ….
   break;        // выход из цикла (значения сохраняются)
   ….
    continue;    // пропуск операторов до конца цикла
   ….

}
```



- При использовании вложенных *if – switch*, *break* относится к ближайшему из операторов. Для корректности использовать *{ }*

Язык С. Циклы. Цикл while

while

Синтаксис: *while (<выражение>) <оператор>;*
эквивалентно
for(; <выражение> ;) <оператор>;

- Цикл с предусловием (минимальное количество повторов = 0)
- Выполняется, пока <выражение> ≠ 0
- В теле цикла можно использовать *break* и *continue*

Примеры:

1) *while (a[i] != 0) s += a[i++];*
2) *while (a[i]) s +=a[i];*
3) *while (s[i++]) ;*
4) *while (1) { … }* *// бесконечный цикл*

Язык С. Циклы. Цикл while

Пример:

Вычислить сумму p = $\sum_{k=1}^{\infty} \frac{1}{k}$

с точностью Ɛ =0.0001

```
int n = 0;              // количество итераций
double s, k = 1., eps = 0.0001;

while (( s = 1. / k ) > eps )
{
   p += s;              // вычисление суммы
   k = k + 1.;
   n++;
}
```

Язык С. Циклы. Цикл do-while

do while

Синтаксис: *do <оператор> while (<выражение>);*

- Цикл с пост-условием
- Выполняется, пока *<выражение>* ≠ 0
- Минимальное количество повторов - 1
- В теле цикла можно использовать *break* и *continue*
- Всегда заканчивается “;”

Пример:

```
do
{
    switch (ch = getchar())
    {
    ….
    }
}
while (ch != ‘q’);          // цикл, пока не введен “q”
```

Язык С. Циклы.

Особенности использования “;”

1)
```
while (a[ i ] != 0) ;// зацикливание
{
    s += a[ i ]; i++;
}
```

2)
```
while(a[i++] != 0) ;
 {
    s += a[ i ]; i++;       // выполняется один раз
 }
```

3)
```
do
 {
…..
 }
 while ( i < 10);     // всегда нужна “;”
```

Язык С. Оператор go to

go to или *goto*

Синтаксис:

go to метка;
….
метка: **….**

- Использование *go to* не рекомендуется
- Метка заканчивается “:” и должна быть уникальной
- Имя метки - идентификатор
- Переходы внутрь тела цикла запрещены

Пример:

….
go to m1;
….
m1:

Язык С. Оператор go to

Примеры допустимого использования *go to*:

1) Выход из вложенных циклов на внешний уровень (использование *break* в этом случае усложняет логику цикла)

2) Передача управления из разных точек программы в одну точку (например, обработка ошибок)

В остальных случаях можно обойтись без использования *go to*.

Пример:

a) *loop:*
…..
…..
go to loop;

b) *while (1)*
{
…..
…..
}

Язык С. Указатели

Указатель: переменная, содержащая адрес другой переменной

Операции с указателями:

`&` - получение адреса (ссылки)

`*` - обращение по адресу (разадресация)

Описание указателей:

```
int *a;          // указатель на переменную int
double *b;       // указатель на double
int *a[5];       // массив из 5 указателей
int (*a)[5];     // указатель на массив
```

- Указатели обязательно должны быть инициализированы, особенно с классом памяти *auto*
- Нулевые указатели запрещены

Язык С. Указатели

Примеры.

```
int a, b;
int *c;

c = &a;   // инициализация указателя
*c = 0;   // эквивалентно  a = 0

c = &b;
*c = 0;   // эквивалентно b = 0
//-------------------------------------------------------------
static int *a;
*a = 5;   // обращение по нулевому адресу

auto int *a;
*a = 5;   // обращение по произвольному адресу
```

Язык С. Указатели. Операции

Адресная арифметика. Допустимые операции:

++ инкремент
-- декремент
+ сложение с константой
- вычитание константы
- разность указателей (результат – число)

```
char *a
short *b;
double *c;
a++;      // a = a + 1
b++;      // b = b + 2
c--;      // c = c - 8
// ------------------------------------------------------------------
b = b + 3;      // b = b + 3 * sizeof(short)
c = c - 4;// c = c - 4 * sizeof(double)
```

Язык С. Указатели и массивы

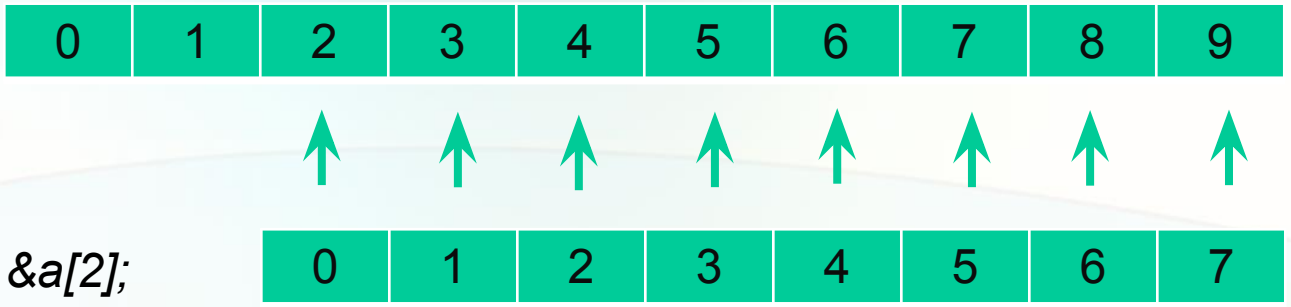
Пример.

```
int a[10], b[20], *p;
p = &a[0];      // или p = a
                // имя массива = адрес его первого элемента
p[3] = 0;       // эквивалентно a[3] = 0

p = &a[2];
p[3] = 0;       // эквивалентно a[5] = 0

p = &b[3];
p[3] = 0;       // эквивалентно b[6] = 0

p--;
p[3] = 0;       // эквивалентно b[5] = 0
```

Имя массива – константа, указатель на 0 элемент массива
Указатель – переменная

Язык С. Указатели и массивы

int a[10];



Пример

```
char str[100];
char *p = str;

while (*p++);        // поиск нулевого элемента в строке
```

Язык С. Указатели и строки

a) *char str[20] = “text string”; // длина строки = 20*

b) *char *str = “text string”; // размер массива = длине строки*
str = “another string” // первая строка «потеряна»

Массивы указателей на строки

```
char str[5][20] =
{
   “text”,
   “another text”,
   “text 3“
}
```

Строки одинаковой длины

```
char *str[5] =
{
   “text”,
   “another text”,
   “text 3“
}
```

Строки разной длины

Язык С. Указатели и строки

```
char *str[4] =
{
    “text”,
    “another text”,
    “text 3“,
    “big text”
}
```