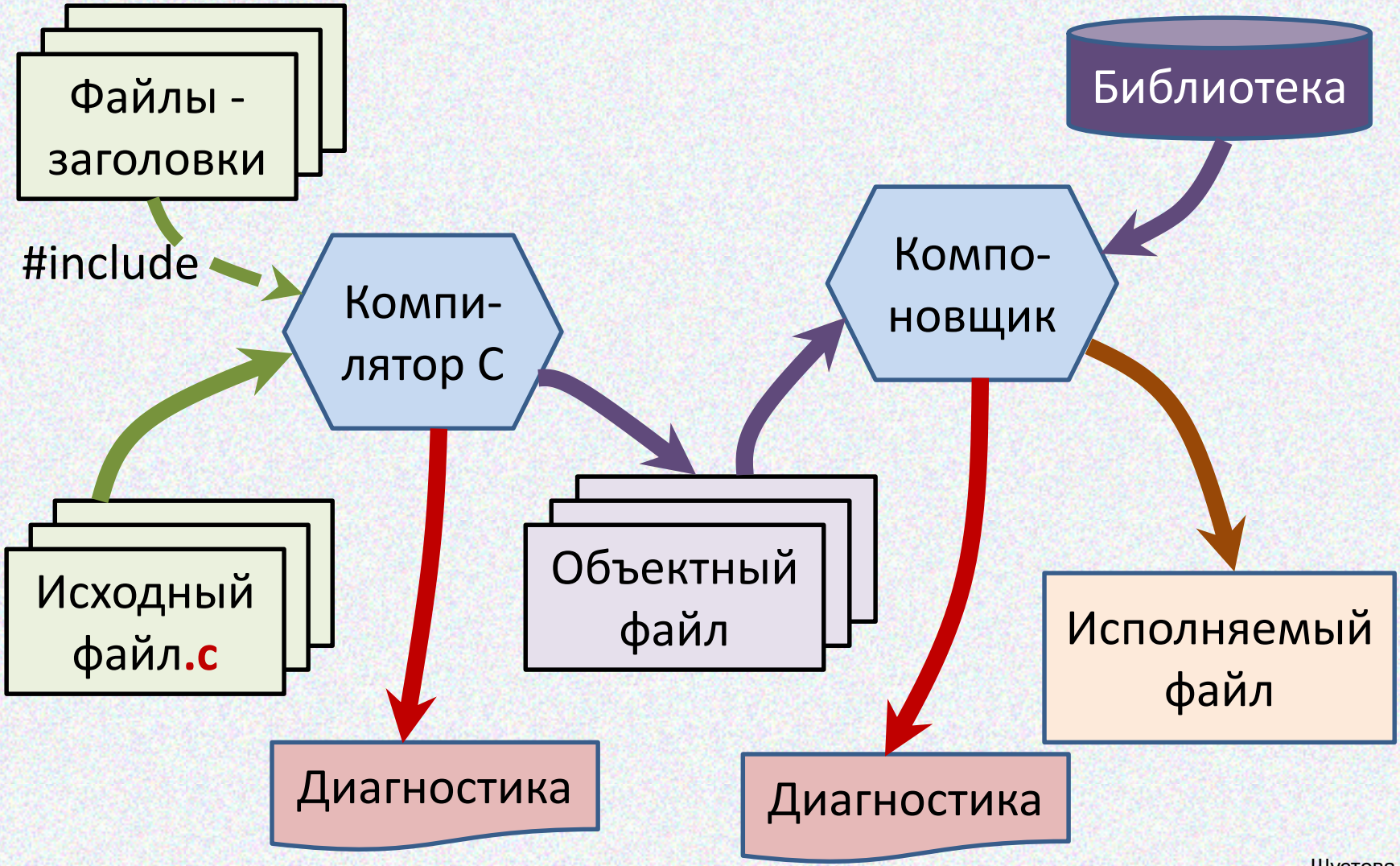
Материалы презентации предназначены для размещения только для использования студентами кафедры «Компьютерные системы и технологии» НИЯУ МИФИ дневного отделения, изучающими курс «Программирование (Алгоритмы и структуры данных)».

С1. Основы языка программирования С

40 слайдов

Шустова Л.И.

Этапы обработки программы



Структура исходного файла

```
// файл.c
#include   <stdio.h>
#include   "myheaderfile.h"
Прототипы функций
Глобальные данные
/*
   комментарий
*/
Определение функции 1
Определение функция 2
...
```

Определение функции

тип имя_функции **(***тип параметр_1, . . .* **)**

{

предложение описания типа_1

. . .

предложение языка_1

. . .

return [*выражение*] **;**

}

Задача 1 (1)

Написать функцию, которая вычисляет $y = x^n$.

Здесь x – вещественное

n – целое

При $n > 0$ $x^n = x \times x \times \ldots \times x$

При $n = 0$ $x^0 = 1$

При $n < 0$ $x^n = 1 \ / \ (x \times x \times \ldots \times x)$, $x \neq 0$

При $n < 0$ и $x = 0$ $x^n = \infty$

Задача 1 (2)

```
#include <float.h>
double  Pow(double x, int n)
{
    int k = n < 0 ? -n : n;
    double  p = 1;
    if (n < 0 && x == 0)
        return DBL_MAX;
    for (; k > 0; --k)
        p = p * x;
    if (n < 0)
        p = 1 / p;
    return p;
}
```

Предложения языка

- Определение данных

- Описание алгоритма
  - простое предложение
    - *пустое*
    - *вычисления*
    - *управления*
  - составное предложение (блок)

Составное предложение

```
{
    предложение_1
    предложение_2
    . . .
}
```

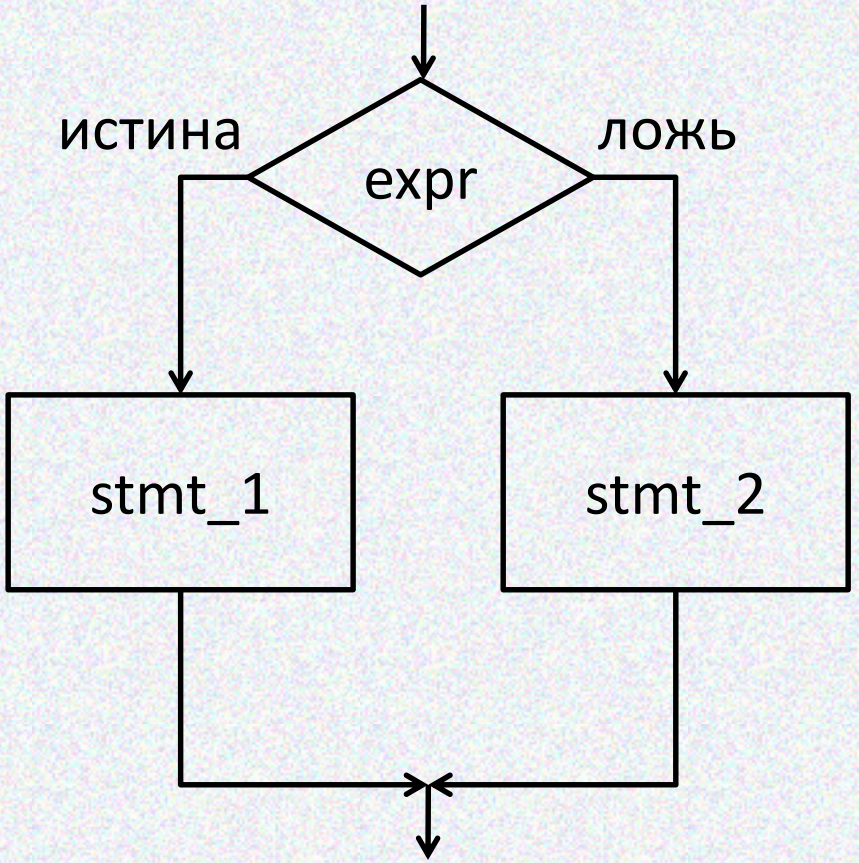
Простое предложение

- Пустое – *;*
- Вычисления – *выражение ;*
- Управления –
  - ветвление (условное, переключатель)
  - цикл (итерационный, параметрический)
  - передача управления:

```
break;
continue;
return [выражение];
```

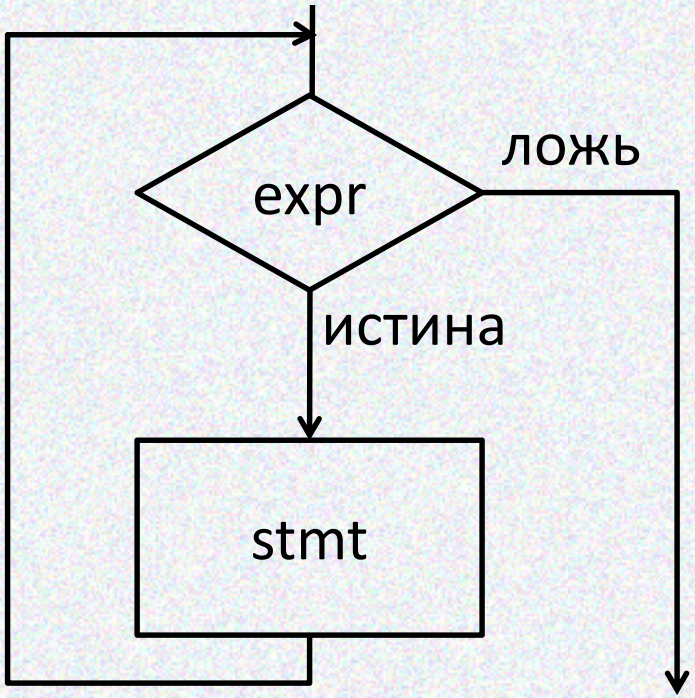
Условное предложение

```
if (expr)
   stmt_1
else
   stmt_2
```

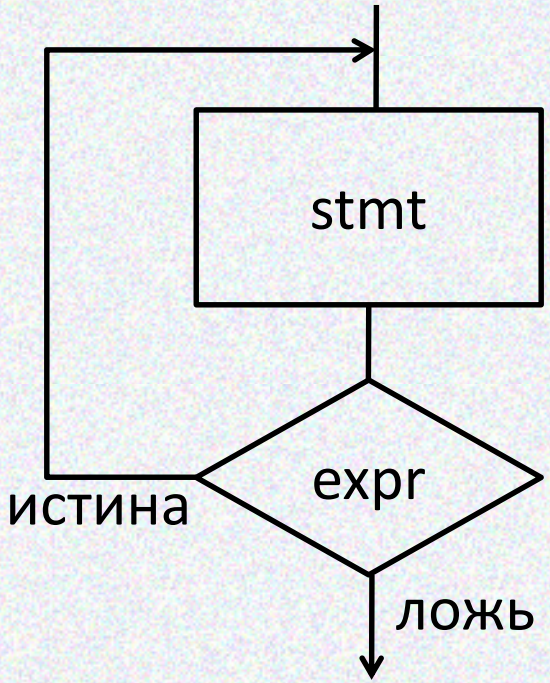


Итерационный цикл

```
while(expr)
    stmt
```




```
do
    stmt
while(expr);
```

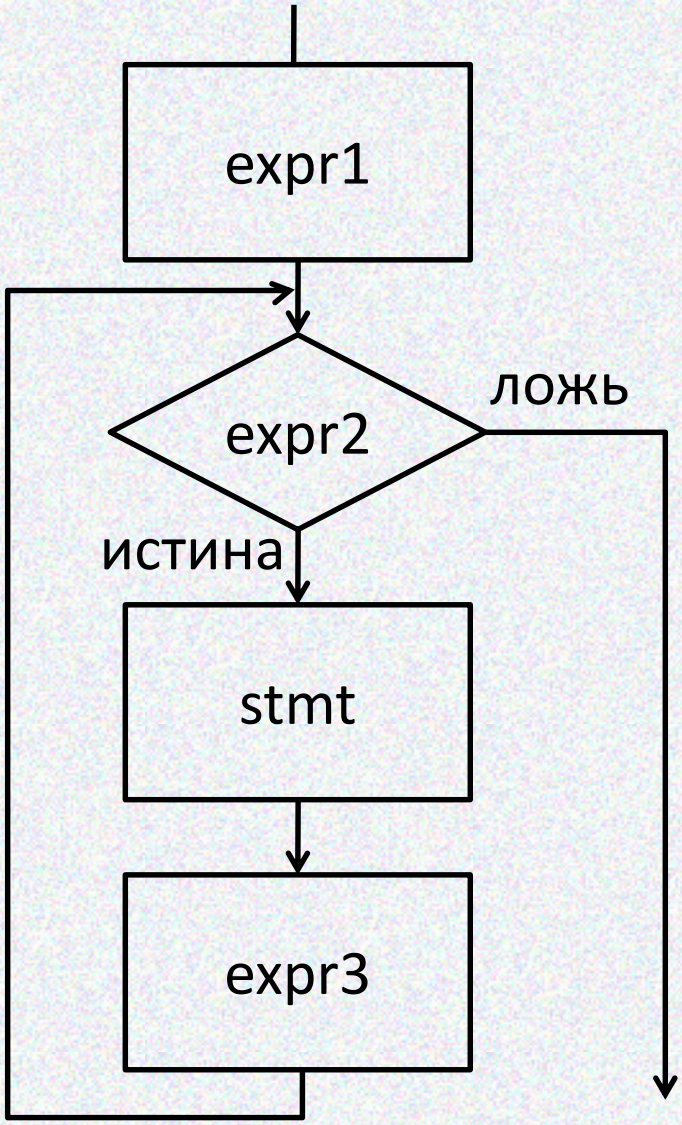


Параметрический цикл

```
for(expr1 ; expr2 ; expr3)
  stmt
```

```
f = 1;
for(i=2;i<=n;++i)
    f=f*i;
```

```
for(;;)
  stmt
```



Определение данных

- Категории данных
  - переменные
  - константы
- Свойства данных
  - тип
  - значение
- Типы данных
  - числа
  - коды
  - адреса

Предложение описания типа

тип имя_объекта [= *инициализация*] , . . . ;

Пример:

```
int a = 5, b;
```

Числовые данные

Типы:

целые

вещественные

Вещественные типы данных:

float – 4 байта

double – 8 байтов

Вещественные константы:

1.25 1.2e-3 1.25E-3 – double

1.25f – float

Целые данные

Целые типы данных — int

модификаторы — long, short

числа без знака — unsigned

Целочисленные константы:

25 25l 25u 25ul

031 029

0x19 0x25fe 0X25FE

Символьные данные

Символьный тип данных

char unsigned char

Символьные константы

'*символ*'

Примеры:

'c' '0' '\377' '\xff' '\XFF'

' ' '\n' '\t' '\'' '\\'

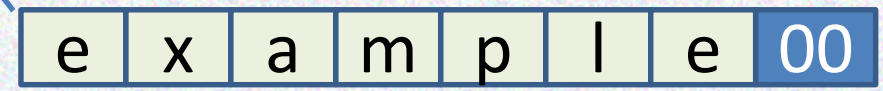
Строковые данные

Строковые константы

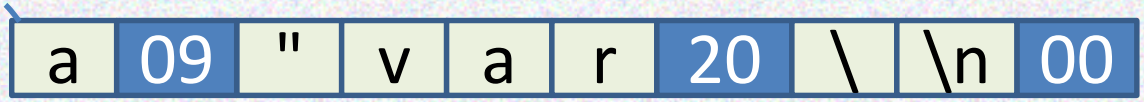
"*произвольный текст*"

Примеры

"example"



"a\t\"var \\\n"



"\377=\xff"

"\39--\x123fe"

Операторы языка (1)

1.	. ->	Выбор члена
2.	[ ] ( ) ++ --	Доступ по индексу Вызов функции Постфиксный инкремент Постфиксный декремент
3.	sizeof ++ -- ~ ! - + & *	Размер объекта или типа Префиксный инкремент Префиксный декремент Дополнение Отрицание Унарный минус Унарный плюс Адрес Разыменование

Операторы языка (2)

4.	* / %	Умножение Деление Остаток о деления (деление по модулю)
5.	+ -	Сложение Вычитание
6.	<< >>	Сдвиг влево Сдвиг вправо
7.	< <= > >=	Меньше Меньше или равно Больше Больше или равно
8.	== !=	Равно Не равно

Операторы языка (3)

9.	&	Побитовое И (AND)
10.	^	Побитовое исключающее ИЛИ (XOR)
11.	|	Побитовое ИЛИ (OR)
12.	&&	Логическое И (AND)
13.	||	Логическое ИЛИ (OR)
14.	? :	Условное выражение
15.	= *= /= %= += -= <<= >>= &= ^= |=	Простое присваивание Операторы типа умножения и присваивание Операторы типа сложения и присваивание Операторы сдвига и присваивание Побитные операторы и присваивание
16.	,	Запятая (последовательность)

Операторы присваивания

= оп=

a оп= выражение → *a = a оп (выражение)*

Правило ассоциативности: справа налево

a = b оп= c = d → *a = (b оп= (c = d))*

выражение	предложение
a = 5	*a* = 5;

Инкремент и декремент

++ --

Префиксный ++a (--a) и *постфиксный* a++ (--a)

Префиксный ++

int a = 2, x; 3

x = ++a; → Результат: a = 3, x = 3

Постфиксный ++

int a = 2, x;

x = a++; → Результат: a = 3, x = 2

Арифметические операторы

унарные	+ –
типа умножения	*
	/
	% (только int)
бинарные (типа сложения)	+ –

Примеры:

5 / 2 → 2

5.0 / 2 → 2.5

5 % 2 → 1

5.0 % 2 → error C2296: '%' : illegal, left operand has type 'double'

Порядок вычисления

a оп b оп c → (a оп b) оп c

Порядок вычисления a, b, c – не определен

Примеры:

int x = 2, z;

z = ++x * x++; → x = 4, z = 9

z = x++ * ++x; → x = 4, z = 9

z = ++x++; → error C2105: '++' needs l-value

z = (++x)++;

Операторы сравнения

< <= >= >

== !=

Примеры:

2 < 5 → результат: 1

2 > 5 → результат: 0

2 < x < 3 → результат: 1, независимо от значения x

2 < x == x < 3 → эквивалентно (2 < x) == (x < 3)

Логические операторы

! && ||

Примеры:

3 < 5 && 5 != 0 → Результат: 1

3 && 5 → Результат: 1

$3 < x < 5$ → Результат: 1

$3 < (x < 5)$ → Результат: 0

Правила вычисления

! && ||

1. Операнды вычисляются в порядке слева направо
2. Вычисления прекращаются, как только результат становится очевидным

Примеры:

a || b && c → ++a || ++b && ++c

a && b || c → ++a && ++b || ++c

Побитные операторы

~ & ^ |

x = 10110010

y = 00010111

~x = 01001101

x&y = 00010010

x^y = 10100101

x|y = 10110111

Условный оператор

expr1 ? *expr2* : *expr3*

```
x = a > b ? a : b;
```

Правило ассоциативности – слева направо

Пример:

expr1 ? *expr2* : *expr3* ? *expr4* : *expr5* →

expr1 ? *expr2* : (*expr3* ? *expr4* : *expr5*)

expr1 ? *expr2* ? *expr3* : *expr4* : *expr5* →

expr1 ? (*expr2* ? *expr3* : *expr4*) : *expr5*

Оператор запятая

expr1 , *expr2* , . . .

Примеры;

```
a = (x = 2, y = 3, x + y)
```

```
for(i = 0, n = 0; i < 10; ++i, n += i)
        ;
```

Ввод/вывод (1)

#include <stdio.h>

Вывод

printf(“*строка_формата*”, *expr1*, *expr2*, …);

Строка формата: произвольный текст + *спецификаци_формата*

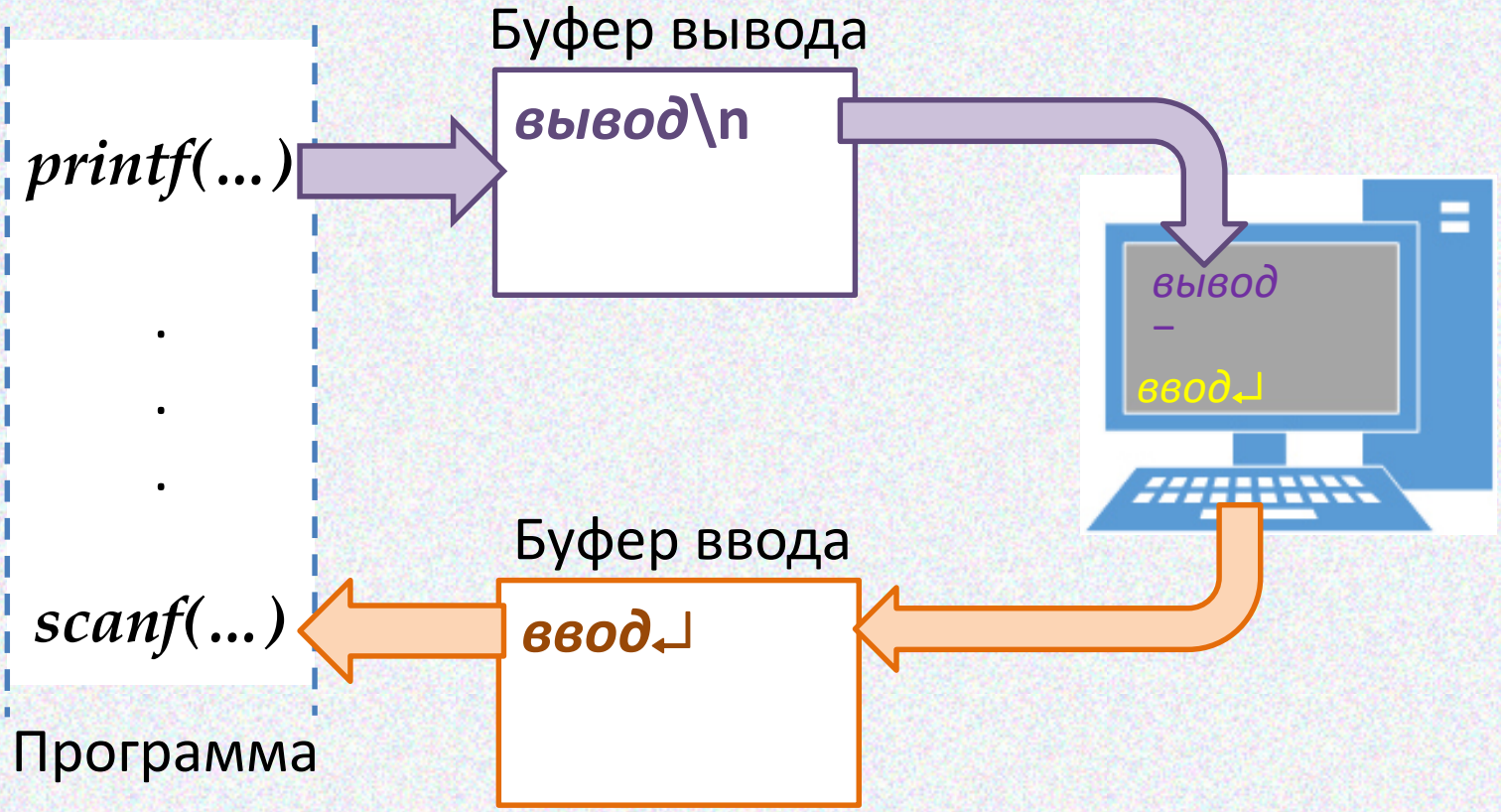
Ввод

scanf(“*спецификации_формата*”, &*var1*, &*var2*, …);

Безопасная версия:

scanf_s(“*спецификации_формата*”, *арг*, *размер*, …,);

Ввод/вывод (2)


Буфер вывода
вывод\n
printf(...)
.
.
.
scanf(...)
Программа
вывод
ввод
Буфер ввода
ввод

Спецификации формата

%c	символы (тип char)
%s	строки символов
%d	целые десятичные числа со знаком (int)
%u	целые десятичные числа без знака
%o	целые восьмеричные числа без знака
%x %X	целые шестнадцатеричные числа без знака
%f %e %E	вещественные числа (float)
%g %G	вещественные числа в стиле d, f или e
%lf %le	вещественные числа (double)
%p	указатель

Задача 2 (1)

Необходимо проверить правильность работы функции возведения вещественного числа в целую степень.

Функция main() должна

- ввести требуемые данные
- вычислить степень числа
- вывести результат

Задача 2 (2)

```
#include <stdio.h>
#include <math.h>
double Pow(double x, int n);
int main()
{
	double x, y, z;
	int n;
	printf("enter x and n: ");
	scanf_s("%lf%d", &x, &n);
	y = Pow(x, n);
	z = pow(x, n);
	printf("pow(%lf, %d) = %lg (Math library: %lg)\n", x, n, y, z);
	return 0;
}
```

Результаты тестирования

```
enter x and n:    2.5   2
pow(2.500000, 2) = 6.25 (Math library: 6.25)
```

```
enter x and n: -1.234567890   0
pow(-1.234568, 0) = 1 (Math library: 1)
```

```
enter x and n: 2.5  -2
pow(2.500000, -2) = 0.16 (Math library: 0.16)
```

```
enter x and n: 0 -2
pow(0.000000, -2) = 1.79769e+308 (Math library: 1.#INF)
```

Задача 3 (1)

Дано целое число n.

Написать функцию, которая определяет количество цифр в записи числа.

Протестировать функцию.

Задача 3 (2)

```
int numOfDigits(int a)
{
    int n = 0;
    do{
        a /= 10;
        ++n;
    } while(a != 0);
    return n;
}
```

Задача 3 (3)

```
#include <stdio.h>
int numOfDigits(int);
int main()
{
    int a, n;
    printf("%s", "Enter integer: ");
    scanf("%d", &a);
    n = numOfDigits(a);
    printf("%d has %d digits\n", a, n);
    return 0;
}
```

Дополнительное задание

1. Дано целое число n. Подсчитать количество двоичных цифр в записи числа. Старшие незначащие нули не учитывать.

2. Дано целое число n. Вывести двоичное представление данного числа. Подсчитать количество единиц в двоичном представлении числа.