

Элементы программы на С

- Описания данных (их типизация),
- Операторы.

1. Пример оператора присваивания:

$y = x + 2 * a;$

Переменная = любое выражение;

это также выражение,
=, +, * - знаки операций

Одноместные операции

$++k \sim k = k+1;$
 $--l \sim l = l-1;$



Это префиксная форма. Выполняется над операндом перед тем, как значение операнда будет использовано во включающем его выражении.

Пример: $x = 5; y = ++x;$

$k++ \sim k = k+1;$
 $l-- \sim l = l-1;$



Это постфиксная форма. Выполняется над операндом уже после того, как значение операнда было использовано во включающем его выражении.

Пример: $x = 5; y = x++;$

Двуместные операции

$+, -, *, /, \% .$



Общеизвестные операции

Пример

$20 \% 6$

результат 2

ы:

(%)

$30 \% 5$

результат 0

Трёхместная операция

$V1 ? V2 : V3 ;$ // V – выражение

Пример:

...

alpha = 3.14 ;

...

betta = 2.5 ;

...

alpha = betta ? 3.14 : 1.57 ;

...

//ответ: alpha = 1.57 ;

//аналог: if betta > 0 alpha = 1.57 else alpha = 3.14 ;

...

Логические операции

$\&\&$ (AND), $\|\|$ (OR), $!$ (NOT)

Пример:

if (x > y && y != 1 || x != 3) z = x ;

Операции

ОТНОШЕНИЯ

<i>Стандартная алгебраическая операция равенства и отношения</i>	<i>Операция равенства или отношения в S_i</i>	<i>Пример условия в S_i</i>	<i>Смысл условия в S_i</i>
Операции равенства			
$=$	$==$	$x == y$	x равен y
\neq	\neq	$x \neq y$	x не равен y
Операции отношения			
$>$	$>$	$x > y$	x больше y
$<$	$<$	$x < y$	x меньше y
\leq	\leq	$x \leq y$	x меньше или равно y
\geq	\geq	$x \geq y$	x меньше или равно y

Операции над указателями (начальные сведения)

1. указатель = &
переменная;

имя типа * имя указателя

любой тип
данных

знак операции
ссылки по
указателю
(обозначение
ссылки)

является
указателем на
переменную типа
имя типа

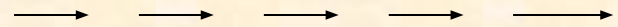
// Пример фрагмента программы

```
...  
Char * ch; //объявление указателей  
Float * p_x;  
...  
p_x = & x; //присвоить адрес x указателю p_x  
...
```

***Еще раз о вычислительных
процессах и
об алгоритмизации***

ТИПОВЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ

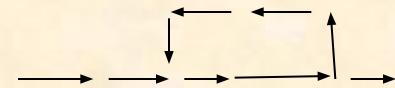
ЛИНЕЙНЫЕ



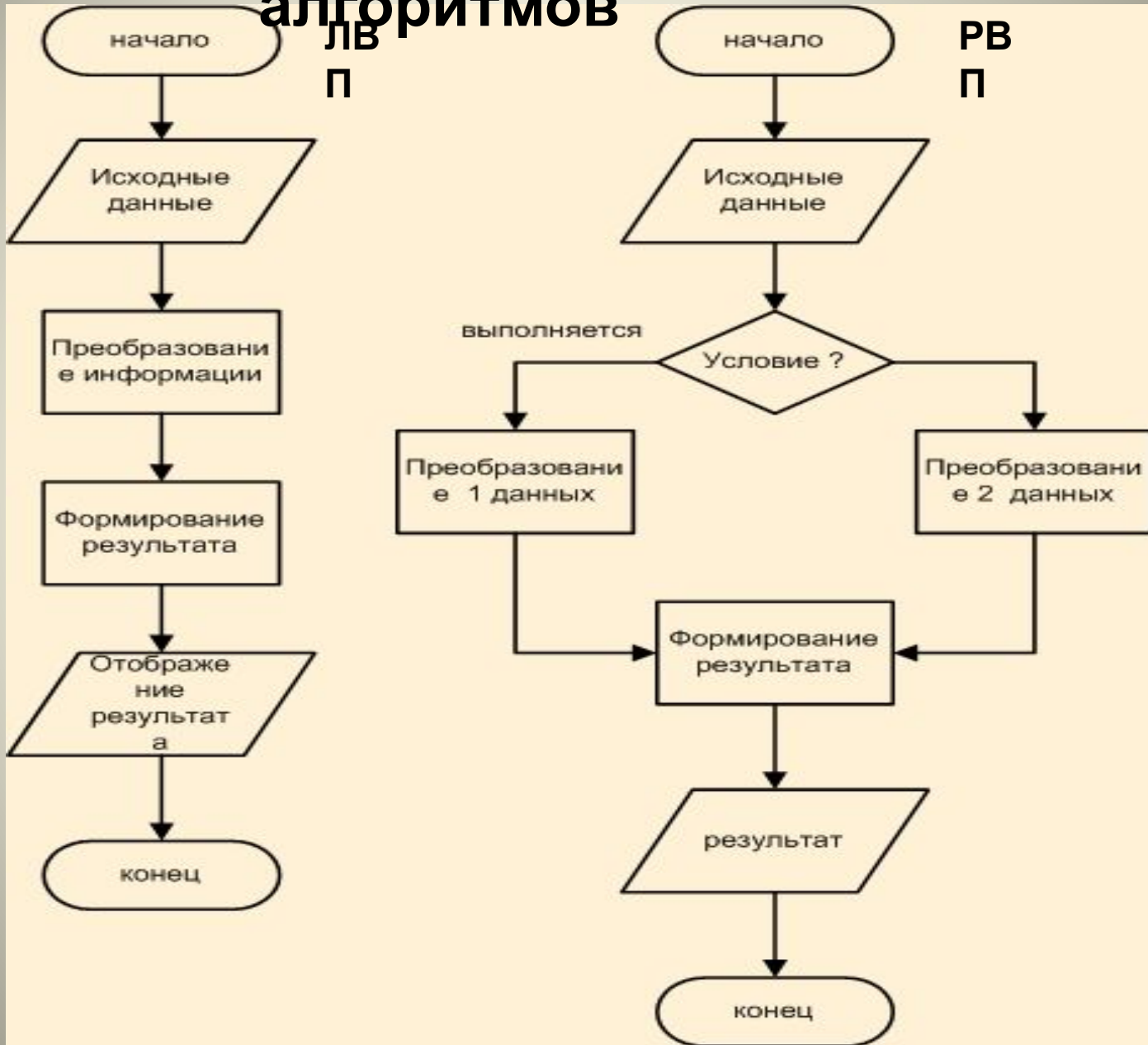
РАЗВЕТВЛЯЮЩИЕСЯ



ЦИКЛИЧЕСКИЕ



Вид блок-схем алгоритмов



ЦВ
П

Цикл for
(Схема организации)



Цикл с предусловием
while



Цикл с постусловием
do...while



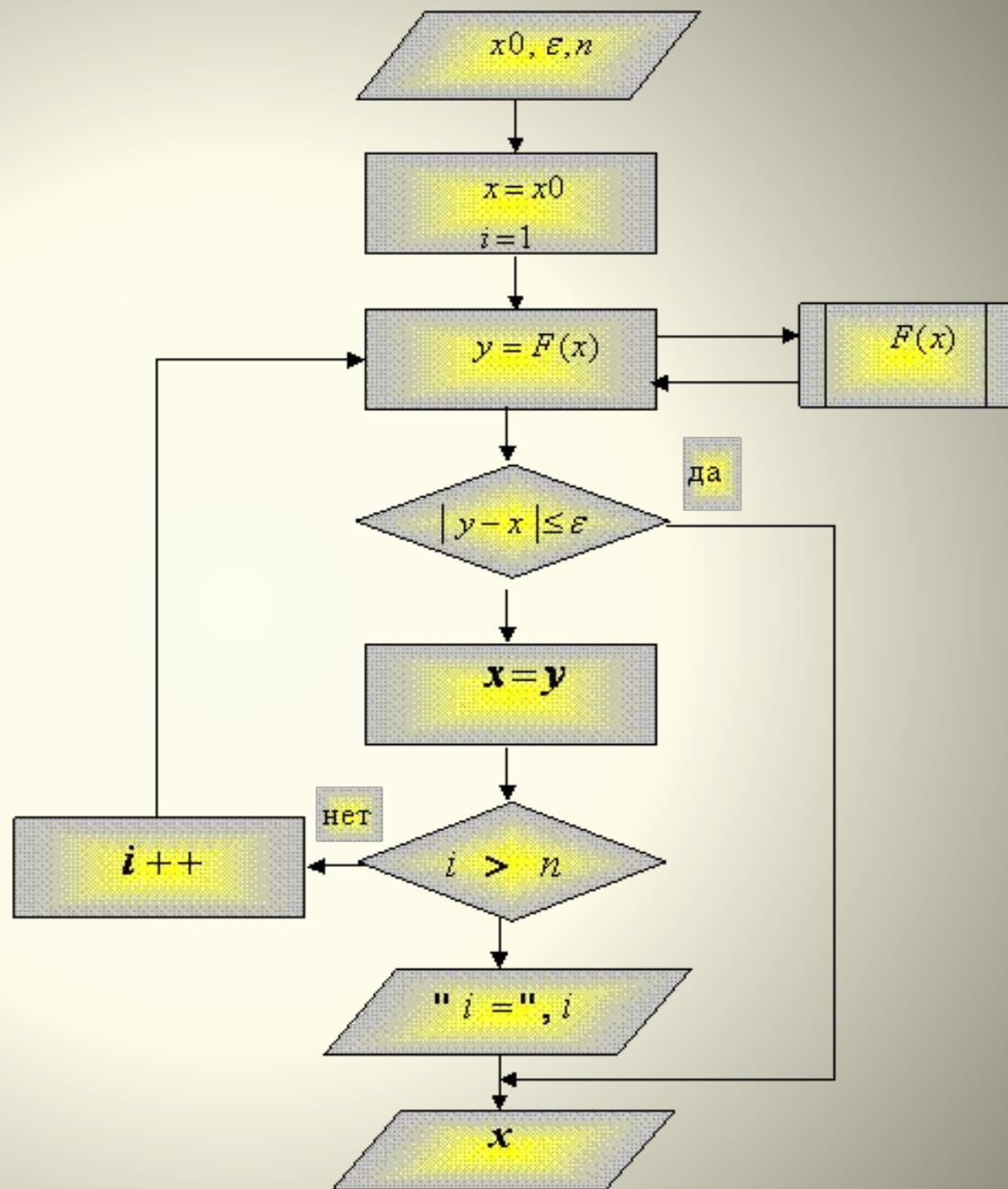
Не типовой ВП

Пример 2:
решение уравнения

$$F(x) = 0,$$

$$x = ?$$

$$x - 2 \ln(x) + \sqrt{x} = 0$$



Понятия, которые необходимо освоить

- алгоритм, алгоритмический процесс, шаг алгоритма ;***
- признаки алгоритма, параметры алгоритма, исходные данные ;***
- элементы блок-схемы ;***
- способы описания алгоритмов, алгоритмический язык, блок схема;***
- классификация алгоритмов, ЛВП, РВП, ЦВП и др.;***
- правила обработки информации, следование, ветвление, цикл;***
- ветви алгоритма, параметр цикла, итерационные циклы, алгоритм проверки изменения исходной величины ;***
- процесс, предопределённый процесс , ручная операция, подготовка, решение, соединитель, терминатор;***
- пояснительный текст, символ комментария, автофигуры, группирование символов блок-схемы ;***
- множество промежуточных результатов, множество окончательных результатов;***
- интерфейс;***
- технология алгоритмизации.***

О типах данных

ПРАВИЛА ТИПИЗАЦИИ ОБЪЕКТОВ В ПРОГРАММАХ

тип имя_переменной;

тип имя_массива [индекс 1] [индекс 2]...[индекс N];

тип имя_функции;

тип указатель на объект;

тип void;

***typedef* тип объект;**

***extern* тип имя_функции;**

БАЗОВЫЕ ТИПЫ ДАННЫХ

char -символьные переменные; в одних реализациях этот тип используется со знаком *signed*, в других – без знака, т.е. с

unsigned

int -целые,

float -с плавающей точкой,

double -с плавающей точкой двойной длины,

long -длинные; они являются, как минимум, 32-битовыми,

short -короткие; они являются 16-битовыми.

- Те элементы данных, которые сохраняют неизменные значения на протяжении всего времени работы программы, принято называть **константами**.
- Другие же объекты, являющиеся предметом изменения в ходе выполнения алгоритма, называют **переменными**.

типы	категория типов		
<i>short, int, long, long long</i> (знаковые и без знаковые)	целые типы	арифметические типы	скалярные типы
<i>char</i> (знаковый и без знаковый)			
<i>_bool</i> (стандарт C99)			
<i>enum { ... }</i>			
<i>float, double, long double</i>	типы с плавающей точкой		
<i>float_complex, double_complex, long_double_complex, float_Imaginary, double_Imaginary, long_double_Imaginary</i> (стандарт C99, для оптимизации)			
<i>T *</i>	Типы указателей		
<i>T [...]</i>	Типы массивов		
<i>struct { ... }</i>	Типы структур		Составные типы
<i>union { ... }</i>	Типы объединений		
<i>T (...)</i>	Типы функций		
<i>void</i>	Тип <i>void</i>		

Тип	Размер в байтах	Интервал изменения	
<i>char</i>	1 (8)	от -128	до 127
<i>unsigned char</i>	1 (8)	от 0	до 255
<i>signed char</i>	1 (8)	от -128	до 127
<i>int</i>	2 (16)	от -32768	до 32768
<i>unsigned int</i>	2 (16)	от 0	до 65535
<i>signed int</i>	2 (16)	от -32768	до 32768
<i>short int</i>	2 (16)	от -32768	до 32768
<i>unsigned short int</i>	2 (16)	от 0	до 65535
<i>signed short int</i>	2 (16)	от -32768	до 32768
<i>long int</i>	4 (32)	от -2147483648	до 2147483647
<i>signed long int</i>	4 (32)	от -2147483648	до 2147483647
<i>unsigned long int</i>	4 (32)	от 0	до 4294967295
<i>float</i>	4 (32)	от 3.4E-38	до 3.4E+38
<i>double</i>	8 (64)	от 1.7E-308	до 1.7E+308
<i>long double</i>	10 (80)	от 3.4E-4932	до 3.4E+4932

Символ после %	Тип, ожидаемый при вводе	Тип аргумента
<i>d, D</i>	Десятичное целое.	<i>int, long int</i>
<i>e, E</i> <i>f</i> <i>g, G</i>	Величина с плавающей точкой из мантиссы и порядка.	Указатель на <i>float</i>
<i>i, I</i>	Десятичное, восьмеричное, шестнадцатеричное целое.	<i>int, long int</i>
<i>o, O</i>	Восьмеричное целое.	<i>int, long int</i>
<i>x, X</i>	Шестнадцатеричное целое без префиксов 0x или 0X.	<i>int</i>
<i>u, U</i>	Десятичное целое без знака.	Указатель на <i>unsigned int, unsigned long int</i>
<i>c</i> <i>es</i>	Символ для прочтения не пробельного символа.	<i>char</i>
<i>s</i>	Символьная строка.	Символьный массив с завершающим '\0'

Есть также некоторые другие %, *n, p*.

- Все данные записываются в компьютере в виде некоторой последовательности битов. **Бит** – это нулевой или единичный разряд в **двоичном числе**.
- В большинстве компьютеров нельзя обратиться к конкретному биту. Можно записывать или читать только машинное слово.
- **Машинное слово** - двоичное число определенной размерности, используемое в основной системе команд компьютера для обработки данных. Размерность слова зависит от вида компьютера. Оно состоит из байтов.
- **Байт** - машинное слово минимальной размерности, адресуемое в процессе обработки данных. Размерность байта - **8 бит** - принята не только для представления данных в большинстве компьютеров, но и в качестве стандарта:
 - для хранения данных на внешних носителях;
 - для передачи данных по каналам связи;
 - для представления текстовой информации. Каждый байт кодирует один символ текста;
 - для определения размерности машинных слов. Размерность машинного слова выбирается кратной байту (см. слайд 16).

Типы данных в программах **(примеры)**

Пример (типов переменных и гл.функции)

```
# include <stdio.h>
int main ( )    //тип главной функции
{
    int a, b, c e; //тип переменных
    printf ("Введите данные: ");
    scanf ("%d%i%o%u%x", &a, &b, &c,
            &d, &e ); //использование типов
    printf ("%d %d %d %d %d\n", a, b, c, d, e);
    return 0;
}
```

```
/* Программа печати массива по строкам и столбцам */
```

```
# include <stdio.h> //прототип функций ввода\вывода  
# include <conio.h> //прототип функции getch() и clrscr ( )  
# define N 4 //N=4 - константа  
void printArray (int[ ][N], int klutch); // Прототип функции  
void main ( ) //описание главной функции (заголовок)  
{  
 int klutch;  
 int array [N][N] = { //инициализация массива  
 {1, 5, 9, 13},  
 {2, 6, 10, 14},  
 {3, 7, 11, 15},  
 {4, 8, 12, 16}};  
 clrscr ( ); //очистка экрана  
 printf ("Печать массива по строкам:\n");  
 printArray (array,1);  
 printf ("Печать массива по столбцам:\n");  
 printArray (array,2);  
}  
void printArray (int A[ ][N], int svitch) //описание функции печати  
 // printArray  
{  
 int i, j;  
 for (i=0; i<=N-1; i++)  
 {  
 for (j=0; j<=N-1; j++)  
 if (svitch==1) printf ("A[%i, %i] = %2i", i, j, A[ i ][ j ]);  
 else printf ("A[%i, %i] = %2i", j, i, A[ j ][ i ]); printf ("\n");  
 }  
}  
} // конец описания функции
```

- Переменным в описаниях можно задавать начальные значения (инициировать). Это можно сделать, вводя инициализирующие выражения непосредственно в инструкции описания переменных

```
char backslash = '\\';  
int i=0; long day = 60*24;  
int ndigit[10]= { 0,0,0,0,0,0,0,0,0,0};  
int mas[] = { 1, 2, 3 };
```

- В последнем примере длина массива определяется по числу инициализирующих значений. Если размер любого типа пропущен, то транслятор определяет его длину, считая инициизирующие значения.

- Для символьных массивов существует специальный способ инициации без скобок и запятых

```
char masch[] = "qwe";
```

- Это сокращение более длинного, но эквивалентного описания

```
char masch[] = { 'q', 'w', 'e', '\0' };
```

- В первом случае длина массива полагается равной количеству символов в строке плюс один, а во втором равна количеству элементов в списке инициализации. В данном конкретном случае размер равен 4 (3 символа плюс закрывающий \0- нуль-символ)
- Еще вариант описания:

```
char strk[80] = "This is a string";
```

объявляет массив, состоящий из 80 элементов типа **char**. 16 из них отличны от нуль-символа

Варианты задач для самостоятельной работы

***1 часть (подобные задачи будут в
контрольных работах и в
тестах)
1 семестра***