# Системное программирование

# Введение

Алгоритм - четко определенный план действий для исполнителя.

Свойства алгоритма:

1. Дискретность - состоит из отдельных шагов.
2. Понятность - должен включать только команды, известные исполнителю.
3. Определенность - при одинаковых исходных данных всегда выдает один и тот же результат.
4. Конечность - заканчивается за конечное число шагов.
5. Массовость - может применяться многократно при различных исходных данных.
6. Корректность - дает верное решение при любых допустимых исходных данных.

Программа это:

1. алгоритм, записанный на каком-либо языке программирования.
2. Набор команд для компьютера.

Команда - описание действий, которые должен выполнить компьютер.

1. Откуда взять исходные данные? (в интернете, исходный код, или пользователь).
2. Что нужно с ними делать? (обработать, преобразовать).
3. Куда поместить результат? (вывести на экран, сохранить в файле).

Машинно-ориентированные (низкого уровня) - каждая команда соответствует одной команде процессора (ассемблер).

Языки высокого уровня - приближены к естественному (английскому) языку, легче воспринимается человеком, не зависят от конкретного компьютера.

1972-1974 - Б. Керниган, Д. Ритчи

Плюсы:

1. Высокая скорость работы программ.
2. Много возможностей.
3. Стал основой многих современных языков (C++, C#, JavaScript, Java, ActionScript, PHP).

Минусы:

1. Много шансов сделать ошибку, которая не обнаруживается автоматически.

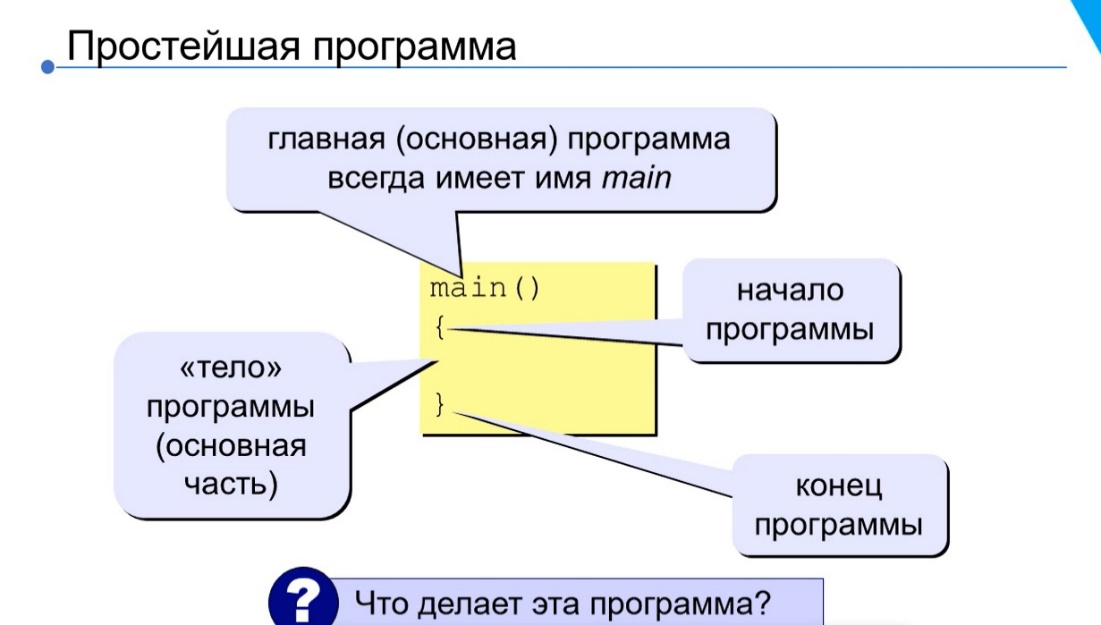
# - директива препроцессора.

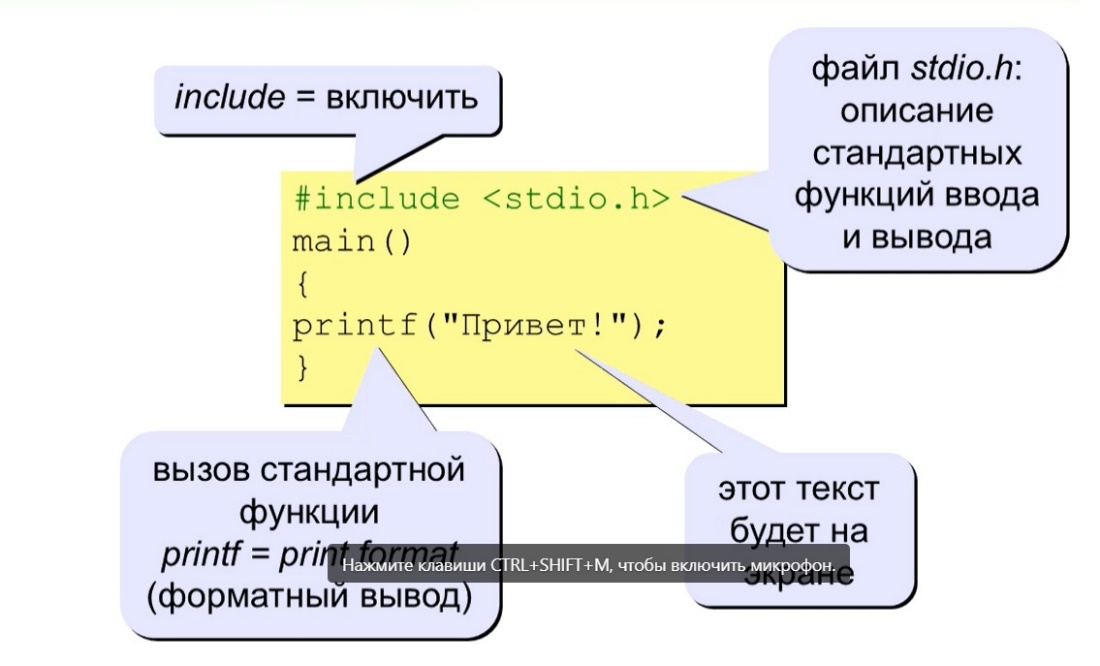
.h - заголовочный файл.

Std - стандарт.

I (input) - ввод.

O (output) - вывод.





; - пустой оператор.

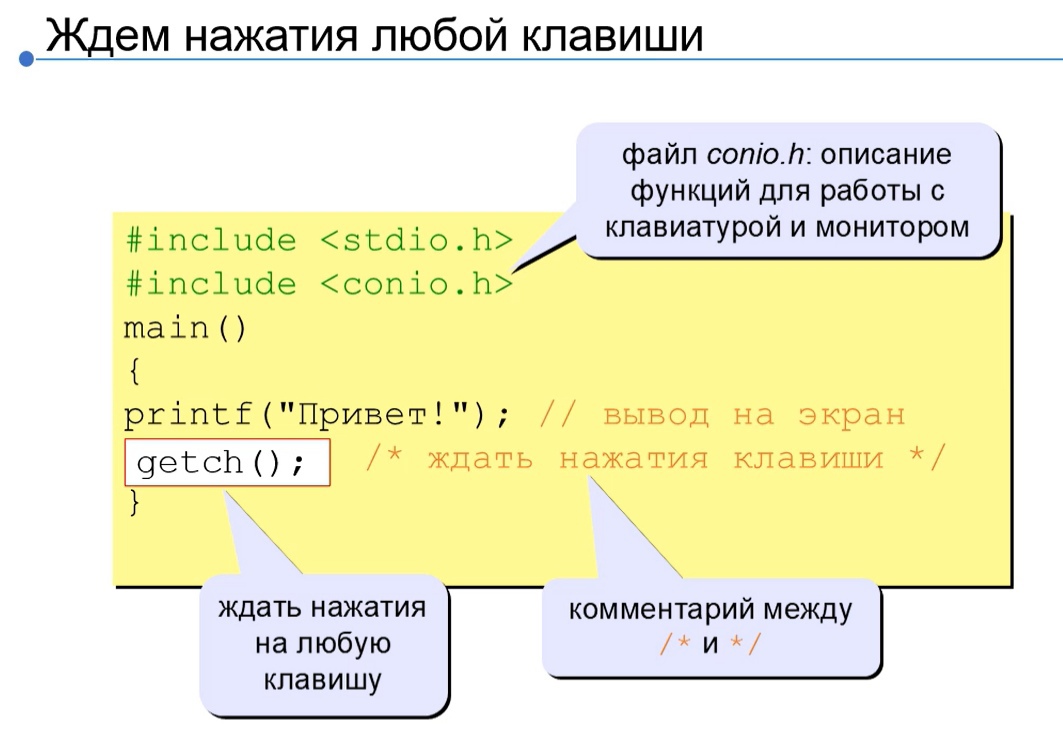
F - формат (txt).

Con - консоль.

Getch (число в милисекундах) - ждать нажатия на любую клавишу (кол-во времени, показывающая файл).

// - однострочный комментарий.

/\* - многострочный комментарий.



Esc последовательности в C:

1. \' - одинарная кавычка
2. \" - двойная кавычка
3. \? - вопросительный знак
4. \\ - обратный слеш
5. \0 - нулевой символ
6. \a - звуковой сигнал
7. \b - backspace
8. \f - перевод страницы - новая страница
9. \n - перевод строки - новая строка
10. \r - возврат каретки
11. \t - горизонтальная табуляция
12. \v - вертикальная табуляция

# Переменные

Переменная - ячейка в памяти компьютера, которая имеет имя и хранит некоторое значение.

1. Значение переменной может меняться во время выполнения программы.
2. При записи в ячейку нового значения старое стирается.

Типы переменных

1. Int - целое число (4 байта)
2. Float - вещественное число, floating point (4 байта)
3. Char - символ, character (1 байт)
4. String - строка

Имена переменных могут включать

1. Латинские буквы (A-Z, a-z)
2. Знак подчеркивания \_
3. Цифры 0-9

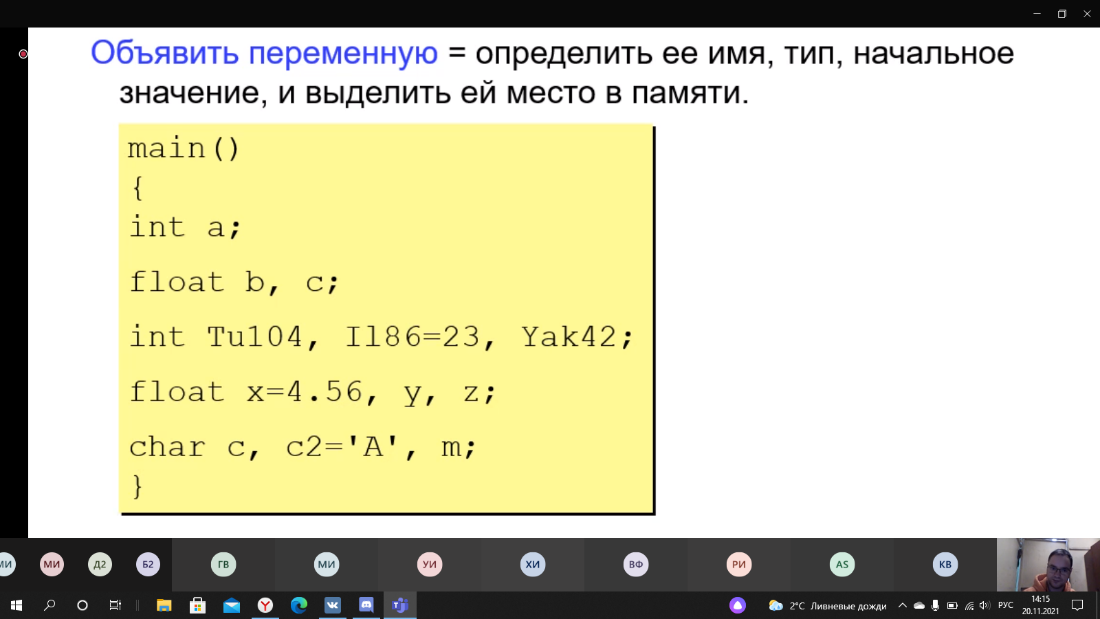
Имя не может начинаться с цифры

Имена переменных не может включать

1. Русские буквы
2. Пробелы
3. Скобки, знаки +. =, !, ? и др.

Объявить переменную = определить ее имя, тип, начальное значение, и выделить ей место в памяти.

Инициализация - объявление и присвоение в одной строке.



Литералы - апостроф.

Если начальное значение не задано, в этой ячейке находиться “мусор”!

Оператор - команда языка программирования высокого уровня.

Оператор присваивания служит для изменения значения переменной.

Общая структура:

Имя переменной = выражение;

Арифметическое выражение может включать

1. Константы (постоянные)
2. Имена переменных
3. Знаки арифметических операций: +, -, \*, /, % (остаток от деления).
4. Вызовы функций
5. Круглые скобки ()

Целая и дробная часть отделяются точкой.

При записи вещественного значения в целую переменную дробная часть будет отброшена.

При делении целых чисел остаток отбрасывается!

|  |  |
| --- | --- |
| Полная запись | Сокращенная запись |
| A = a +1; (инкремент) | A++; |
| A = a + b; | A +=; |
| A = a – 1; (декремент) | A--; |
| A = a – b; | A -= b; |
| A = a \* b; | A \*= b; |
| A = a / b; | A /= b; |
| A = a % b; | A %= b; |

Порядок выполнения операций

1. Вычисление выражений в скобках
2. Умножение, деление, % слева направо
3. Сложение и вычитание слева направо

# Ввод и вывод

#include <stdio.h>

#include <conio.h>

main()

{

int a, b, c;

printf("Введите два целых числа\n");

scanf ("%d%d", &a, &b);

c = a + b;

printf("%d", c);

getch();

}

& - амперсанд, взятие адреса ячейки памяти ОС.

Scanf - форматный ввод.

Формат – символьная строка, которая показывает, какие числа вводятся (выводятся).

%d – целое число

%f – вещественное число

%c – 1 символ

%s – символьная строка

NULL - неопределенность.

Void - пусто.

Минимальное число позиций равно 6

Домашняя работа

#include <stdio.h>

#include <conio.h>

main ()

{

float a, b, c;

printf ("Введите три числа\n");

scanf("%f%f%f", &a, &b, &c);

printf("\n%f + %f + %f = %f", a, b, c, a + b + c);

printf("\n%f \* %f \* %f = %f", a, b, c, a \* b \* c);

printf("\n(%f + %f + %f) / 3 = %f", a, b, c, (a + b + c) / 3);

getch ();

return 0;

}

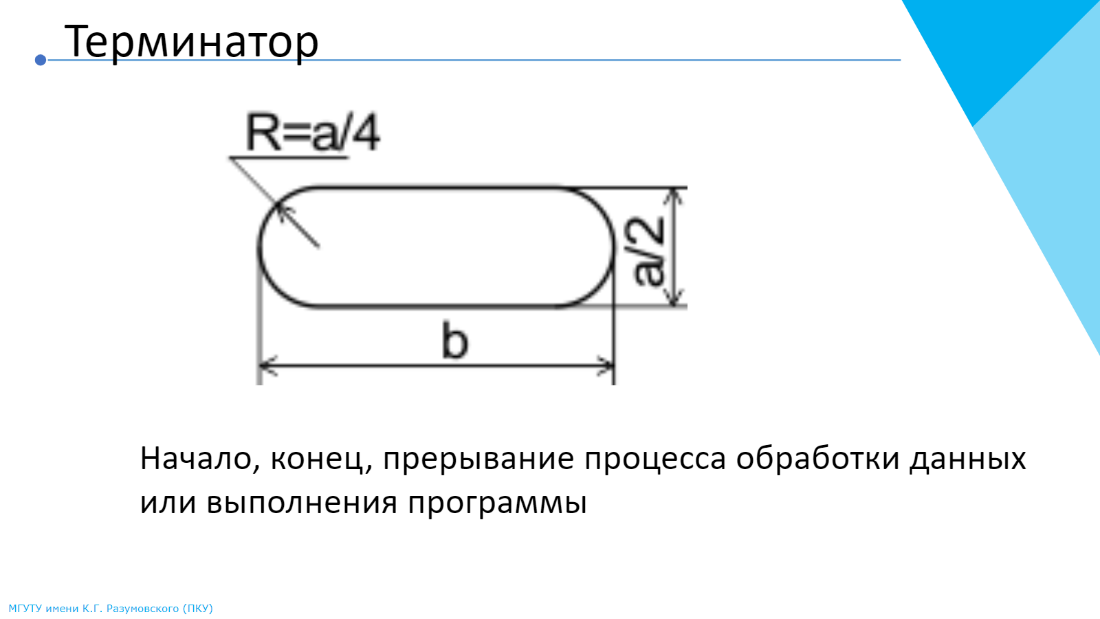
# Основные размеры блок-схем

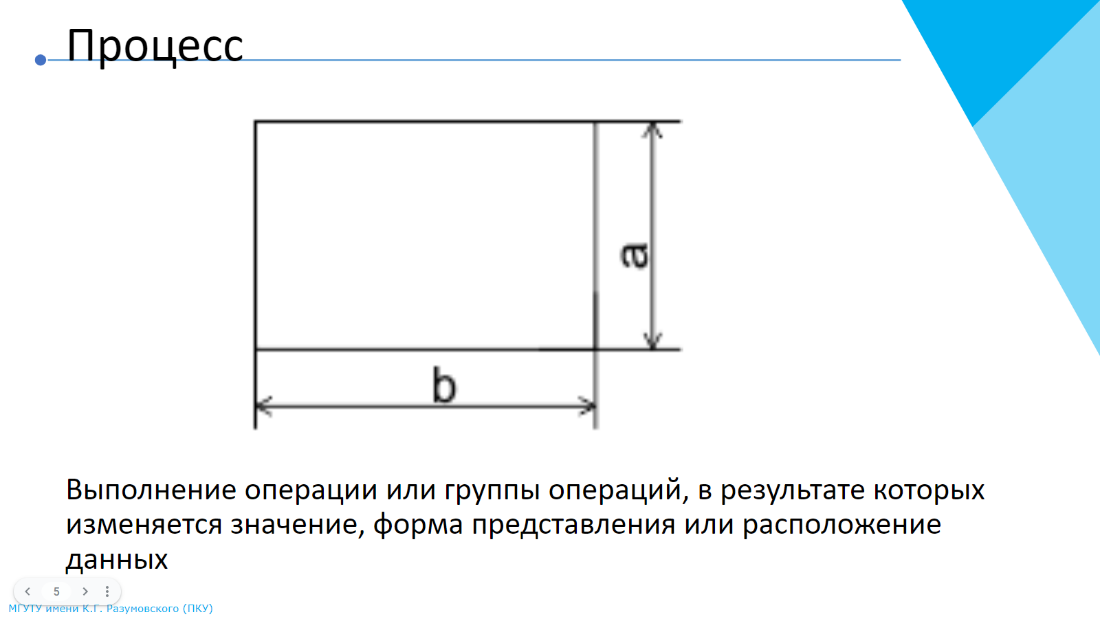
Размер a должен выбираться из ряда 10, 15, 20 мм. Допускается увеличивать размер a на число, кратное 5 мм. Размер b равен 1,5a.

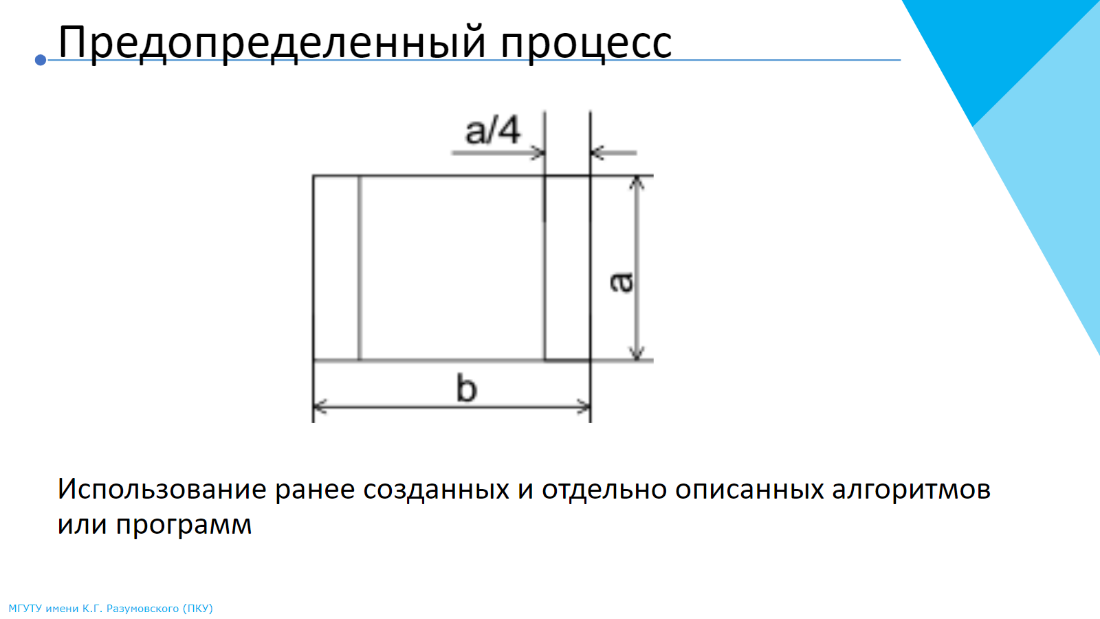
Основным направлением потока в схемах алгоритмов принято направление сверху-вниз, слева-направо.

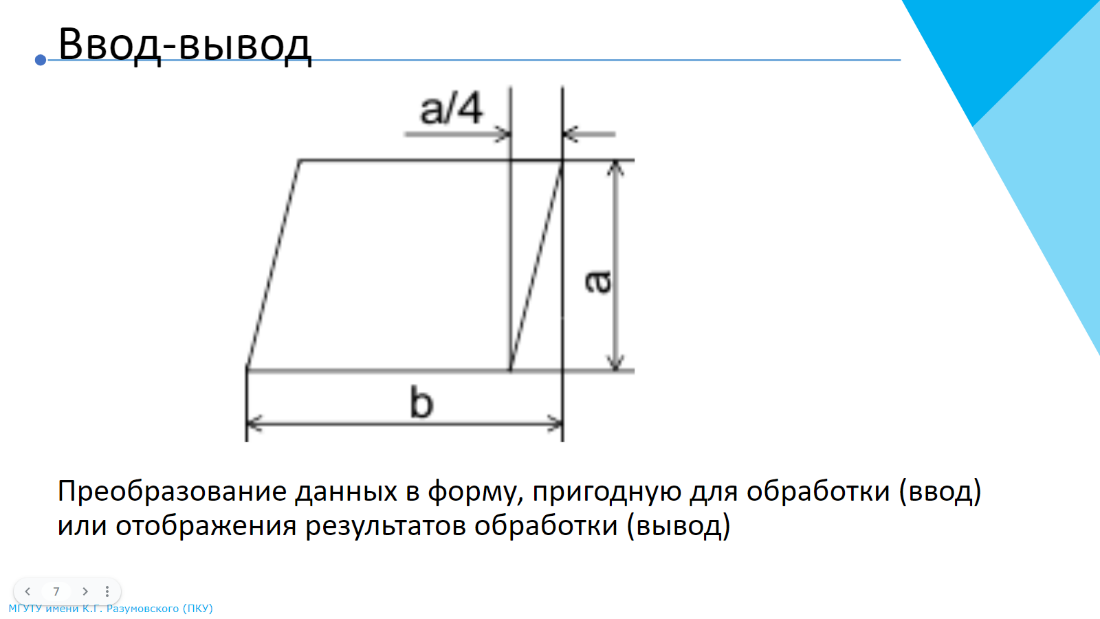
Записи внутри символа должны быть представлены так, чтобы их можно было читать слева направо и сверху вниз, независимо от направления потока.

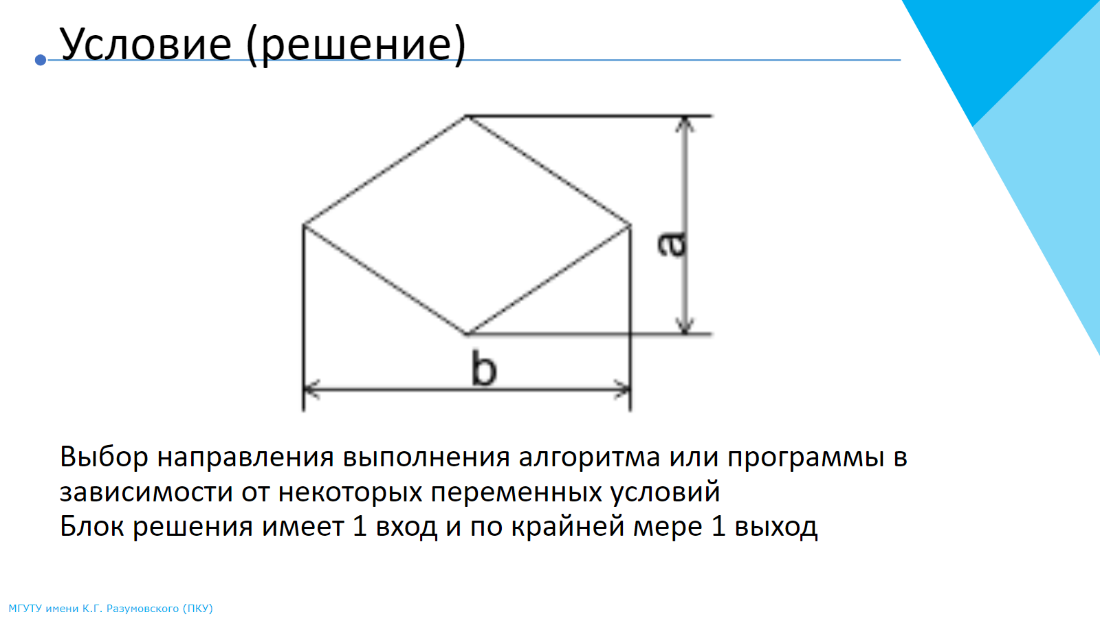
Описание символа должно помещаться справа над символом.

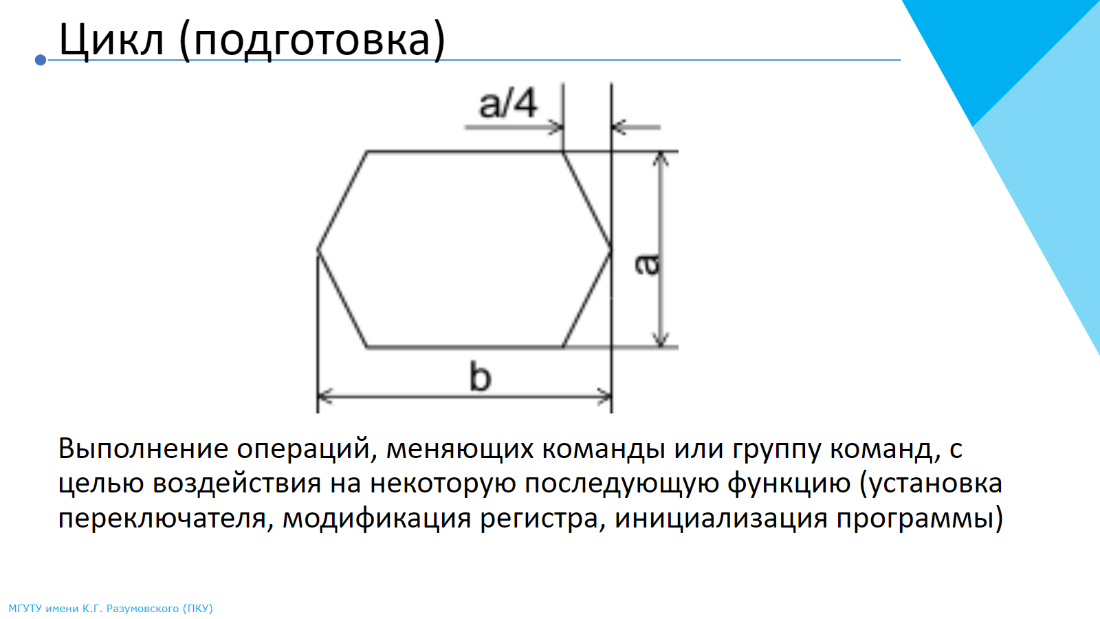


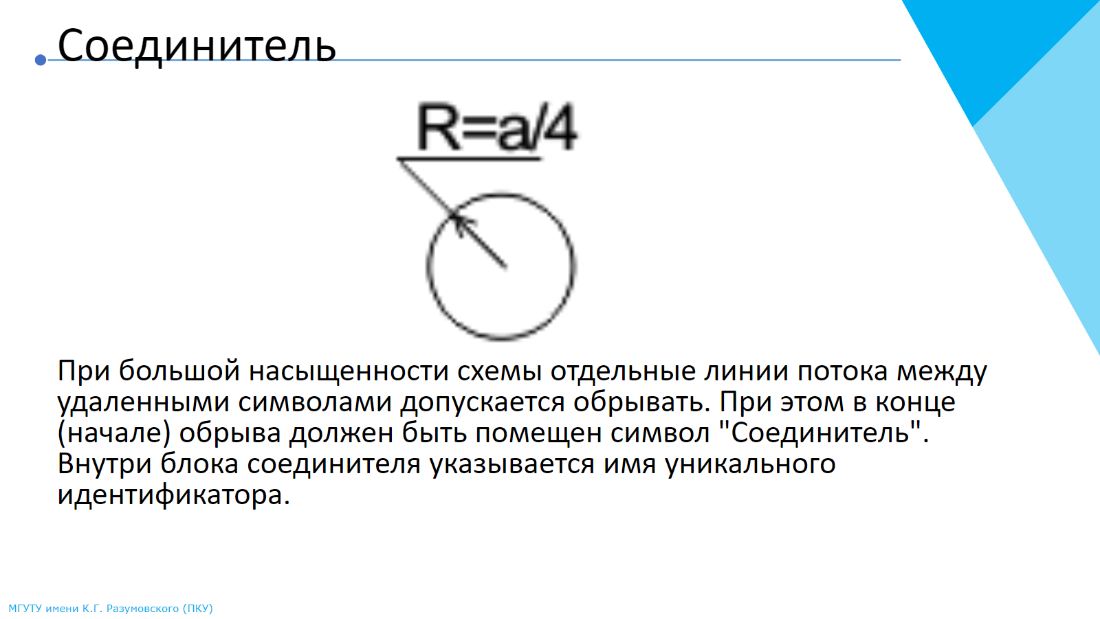






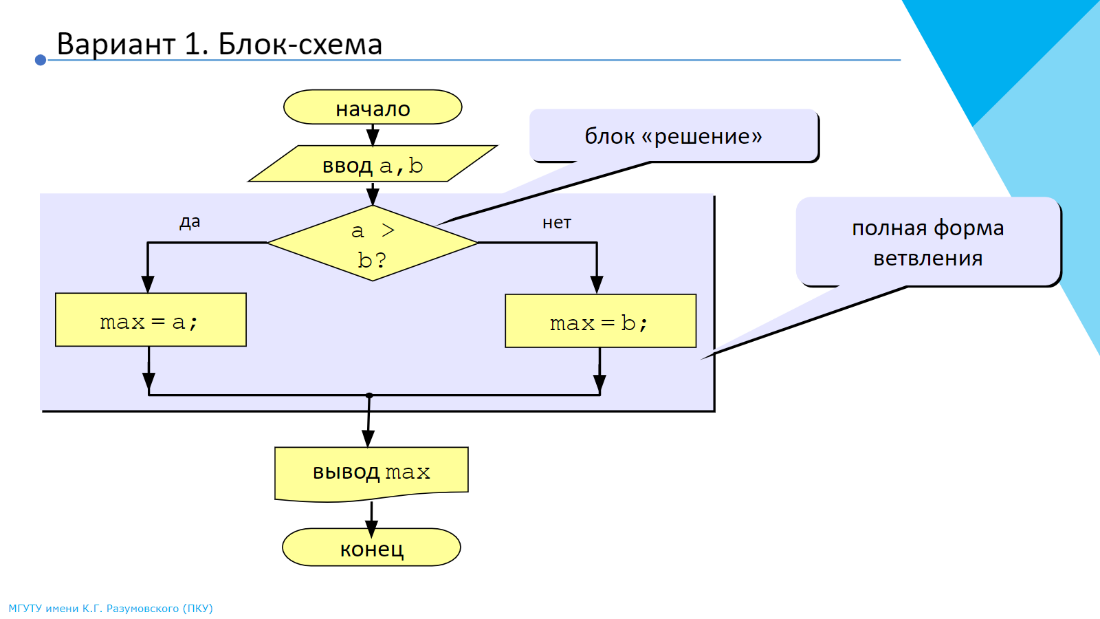






# Ветвление

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются разветвляющимися.



#include <stdio.h>

#include <conio.h>

int main()

{

int a, b, max;

printf("Введите два целых числа:\n");

scanf("%d%d", &a, &b);

If (a > b) {

max = a;

}

else {

max = b;

}

printf("Наибольшее число: %d", max);

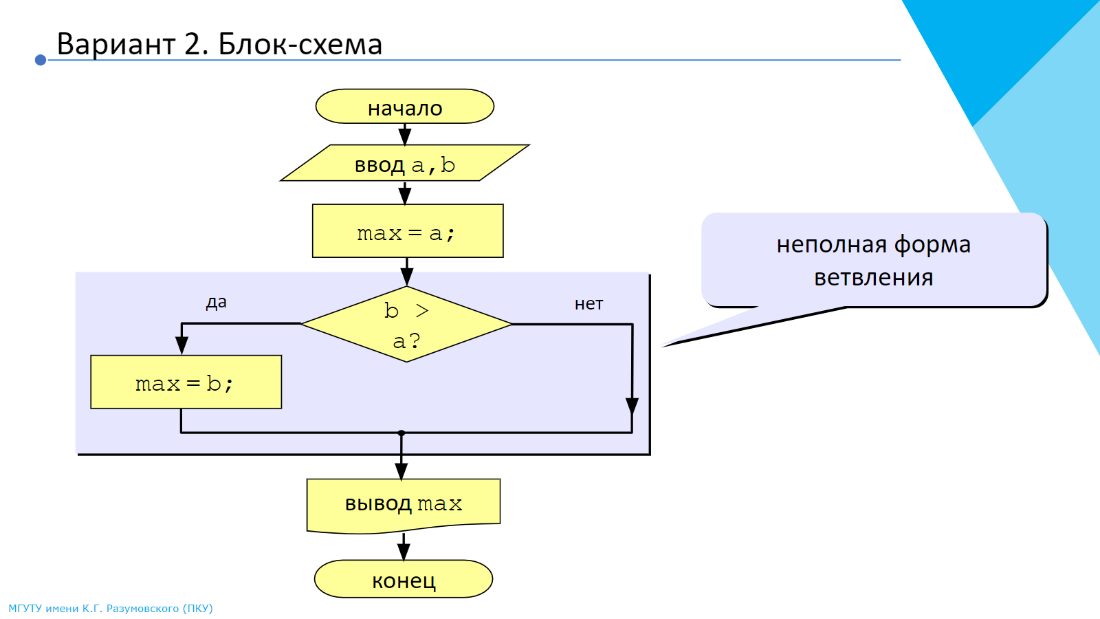
getch();

return 0;

}

Особенности условного оператора:

1. Вторая часть (else …) может отсутствовать (неполная форма)
2. Если в блоке один оператор, то можно убрать {}



#include <stdio.h>

#include <conio.h>

int main()

{

int a, b, max;

printf("Введите два целых числа:\n");

scanf("%d%d", &a, &b);

max = a;

if (b>max)

max = b;

printf("Наибольшее число: %d", max);

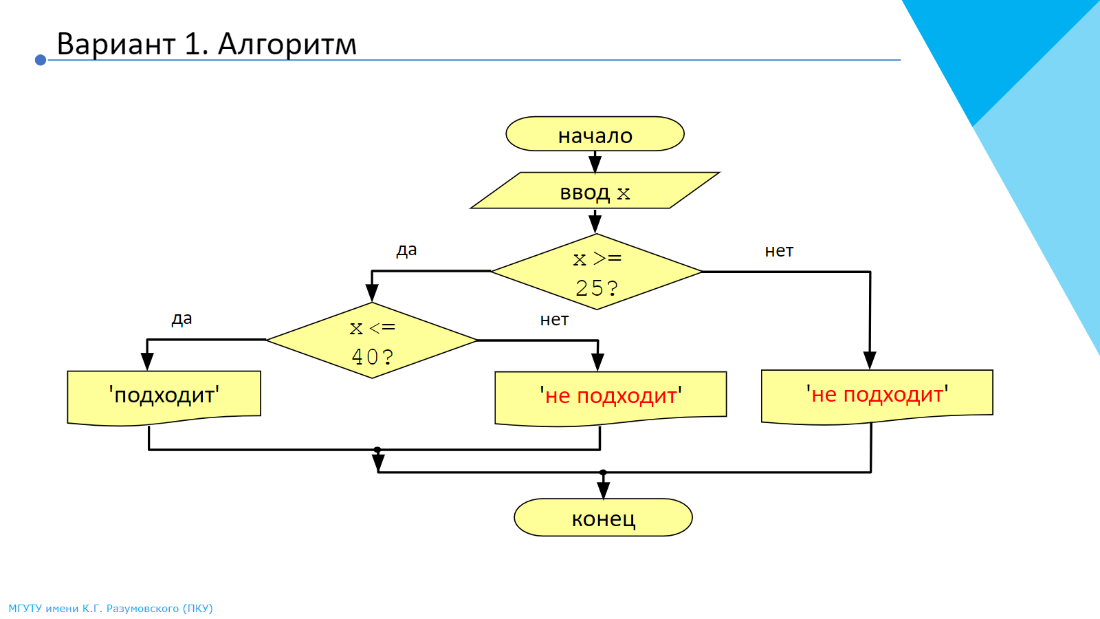
getch();

return 0;

}

# Сложные условия

Фирма набирает сотрудников от 25 до 40 лет включительно. Ввести возраст человека и определить, подходит ли он фирме (вывести ответ «подходит» или «не подходит»).



main()

{

int x;

printf("Введите возраст\n");

scanf("%d", &x);

if (x >= 25)

if (x <= 40)

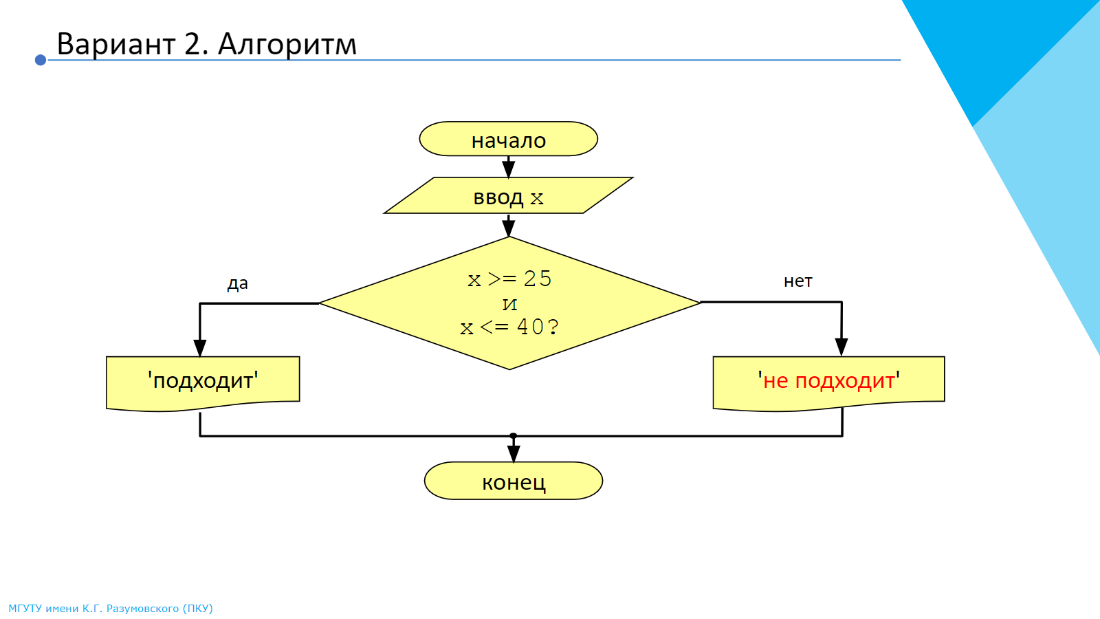
printf("Подходит");

else printf("Не подходит");

else

printf("Не подходит");

}



main()

{

int x;

printf("Введите возраст\n");

scanf("%d", &x);

if ( x >= 25 && x <= 40 )

printf("Подходит");

else printf("Не подходит");

}

Сложное условие – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью логических операций:

! – НЕ (not, отрицание, инверсия)

&& – И (and, логическое умножение, конъюнкция,  
 одновременное выполнение условий)

|| – ИЛИ (or, логическое сложение, дизъюнкция,  
 выполнение хотя бы одного из условий)

Простые условия (отношения)

< <= > >= == (равно) != (не равно)

Порядок выполнения сложных условий:

выражения в скобках

1. ! (НЕ, отрицание)
2. <, <=, >, >=
3. ==, !=
4. && (И)
5. || (ИЛИ)

Домашняя работа

#include <stdio.h>

#include <conio.h>

#include <clocale>

int main ()

{

setlocale(0, "rus");

int a;

printf ("Введите номер месяца:\n");

scanf("%i", &a);

if (a==3 || a == 4 || a == 5)

{

printf ("весна\n");

}

else if (a == 6 || a == 7 || a == 8)

{

printf ("лето\n");

}

else if (a == 9 || a == 10 || a == 11)

{

printf ("осень\n");

}

else if (a == 1 || a == 2 || a == 12)

{

printf ("зима\n");

}

else if (a < 1 || a > 12)

{

printf("Нет такого месяца");

}

getch ();

}

Зачет

Пользователь вводит 2 числа. Найти их разницу и вывести на экран среднее значение этих 3 чисел.

#include <stdio.h>

#include <conio.h>

#include <clocale>

#include <math.h>

int main ()

{

setlocale(0, "rus");

float a, b, c, d;

printf ("Введите два числа\n");

scanf("%f%f", &a, &b);

if (a > b) {

c = a - b;

d = (a + b + c)/3;

printf("c = %f\n", c);

printf("d = %f\n", d);

}

else printf("попробуйте еще раз\n");

getch ();

return 0;

}

# Циклы

Циклы – многократное выполнение одинаковой последовательности действий.

* Цикл с известным числом шагов (цикл со счетчиком).

for (*начальные значения*;

*условие продолжения цикла*;

*изменение на каждом шаге*)

{

// тело цикла

}

Особенности:

1. Условие проверяется в начале очередного шага цикла, если оно ложно цикл не выполняется;
2. Изменения (третья часть в заголовке) выполняются в конце очередного шага цикла;
3. Если условие никогда не станет ложным, цикл может продолжаться бесконечно (зацикливание).

* Цикл с неизвестным числом шагов (цикл с условием)

while (условие)

{

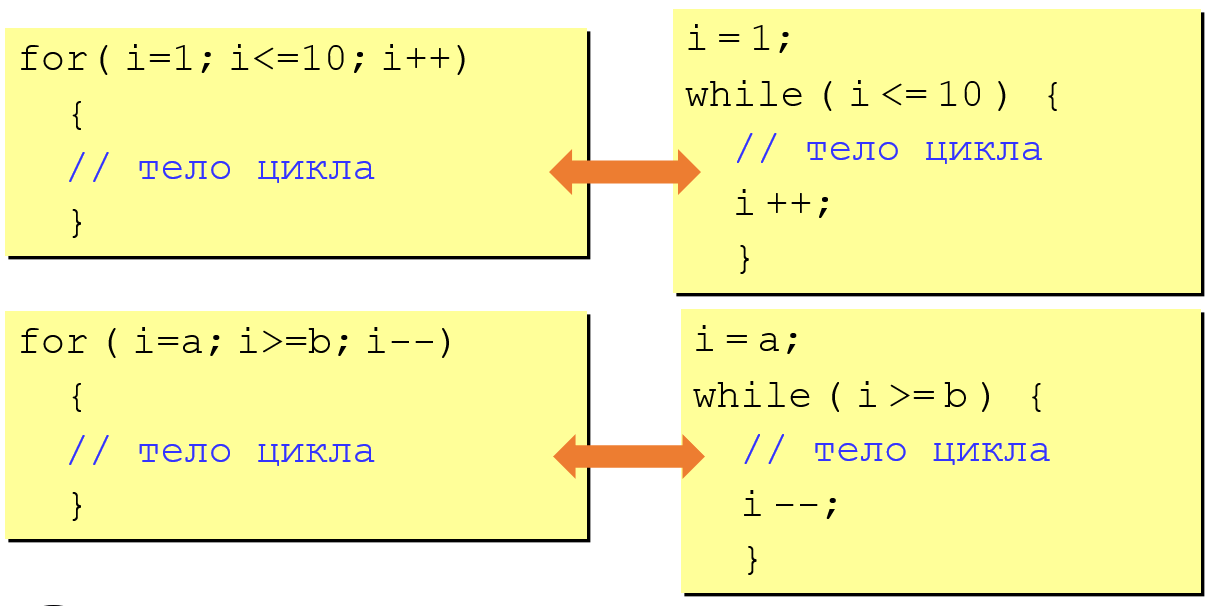
// тело цикла

}

Особенность:

1. Можно использовать сложные условия;
2. Если в теле цикла только один оператор, скобки {} можно не писать;
3. Условие пересчитывается каждый раз при входе в цикл;
4. Если условие на входе в цикл ложно, цикл не выполняется ни разу;
5. Если условие никогда не станет ложным, программа зацикливается.

Замена цикла for на while и наоборот



Предусловие

While (условие)

{…}

Постусловие

Do

{…}

While (условие);

Домашнее задание

№1

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

float b,c,z,a,s;

s = 1;

b = 1;

c = 2;

z = -1;

a = z \* (b/c);

while (fabs (a) > 0.001)

{

s = s + a;

z = z \* -1;

b = b + 1;

c = 2 \* c;

a = z \* (b/c);

}

printf("s = %f\n", s);

getch();

return 0;

}

№ 2

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

float b,c,z,a,s;

s = 0;

b = 1;

c = 2;

z = -1;

a = 1;

while (fabs (a) > 0.001)

{

s = s + a;

a = z \* (b/c);

z = - z;

b = b + 1;

c = 2 \* c;

}

printf("s = %f\n", s);

getch();

return 0;

}

№3

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

float a,b,c,z,d,s;

s = 0;

b = 2;

d = 3;

c = 3;

z = 1;

a = 1;

while (fabs (a) > 0.001)

{

s = s + a;

a = z \* (b/(c\*d));

d = d \* 3;

z = -z;

b = b + 2;

c = 2 + c;

}

printf("s = %f\n", s);

getch();

return 0;

}

№4

#include <stdio.h>

#include <conio.h>

#include <math.h>

int main()

{

float a,b,c,z,d,g,s;

s = 0;

b = 2;

d = 3;

c = 2;

z = 1;

a = 1;

g = 1;

while (fabs (a) > 0.001)

{

s = s + a;

a = z \* (b/(c\*d));

d = d \* 3;

z = -z;

b = b + 2;

c = g + c;

g = g + 1;

}

printf("s = %f\n", s);

getch();

return 0;

}

Расчет Пасхи

#include <stdio.h>

int main()

{

int y, a, b, c, d, e, f;

printf("Enter a year ");

scanf("%i", &y);

a = y % 19;

b = y % 4;

c = y % 7;

d = (19 \* a + 15) % 30;

e = (2 \* b + 4 \* c + 6 \* d + 6) % 7;

f = d + e;

if (f<=26)

{

f = 4 + f;

printf("%d April\n", f);

}

if (f>26)

{

f = f - 26;

printf ("%d May\n", f);

}

return 0;

}

# Массив

Массив - совокупность однотипных элементов, объединенных общим именем.

Массивы:

1. Статические – не меняет значения.
2. Динамические – изменяет значение во время работы.

- new

- realloc

- malloc

- calloc

int M[10], N[7], K[9];

for (int i=0; i<10; i++)

scanf(“%d”, &M[i]);

for (int i=0; i<7; i++)

N[i]=rand()%4-9;

for (int i = 0; i<9; i++)

K[i] = i++\*--i\*2;

Динамические

Printf(“введите n”);

Scanf(“%d”, &n);

Int \* (указатель) M = new int[n];

Delete []M

Синтаксис

‘указатель = тип [размер];

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main()

{

srand(time(NULL));

int \*M, n, imax = 0, imin = 0;

printf("введите кол-во элементов\n");

scanf("%d", &n);

M = new int [n];

for (int i=0; i<n; i++)

{

M[i] = rand () % 50 - 9;

printf("%d", &M[i]);

}

int max = M[0], min = M[0];

for (int i=1; i<n; i++)

{

int max = min = M[0];

if (M[i] > max)

{

max = M[i];

imax = i;

}

if (M[i] < min)

{

min = M[i];

imin = i;

}

}

int temp = M[imin];

M[imin] = M[imax];

M[imax] = temp;

for (int i=0; i < n; i++)

printf("%d", &M[i]);

delete []M;

return 0;

}

Домашняя работа

- calloc

void \*calloc (size\_t num, size\_t size);

- realloc

void\* realloc (void\* ptr (ранее выделенный участок памяти), size\_t size);

- malloc

void \* malloc (size\_t size);

free – освобождает память.

Даны 2 массива: A(n) и B(n).

Все нулевые элементы заменить на сумму max из массива A и B.

Функции

Тип имя (список формальных параметров)

{

Тело функции

}

Где:

Тип – тип результата;

Имя – имя функции;

Список формальных параметров – содержит как имена параметров, так и их спецификации.

Параметры:

- формальные (имена, которые не будем использовать в программе);

- фактические (реальные имена или значения, которое мы используем в программе).

//функция возвращает значение целого типа

int max (int x, int y)

{if (x> y)

Прототип функции состоит только из заголовка:

Тип имя (список формальных параметров)

* При описании прототипов функции список формальных параметров может содержать только спецификации параметров (int max (int, int), для выделения памяти);
* Прототип должен быть обязательно если описание функции стоит ниже

Домашняя работа

Найти все элементы, которые по модулю больше введенного числа; модифицировать его так, чтобы все эти числа были поставлены в первую позицию; все остальные числа увеличить на 2 и поставить на последнюю позицию.

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <time.h>

int main()

{

srand(time(NULL));

int\* A, n, t;

printf("Введите n \n");

scanf("%i", &n);

if (n < 1) printf("Введите положительное чило\n");

else

{

for (int i = 0; i < n; i++)

{

A[i] = rand() % 20;

printf(" %i", A[i]);

}

printf("\n");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (fabs(A[i]) > n)

{

t = A[j];

A[j] = A[i];

A[i] = t;

}

}

}

for (int i = 0; i < n; i++)

{

if (fabs(A[i]) < n) A[i] += 2;

printf(" %i", A[i]);

}

}

return 0;

}

|  |  |
| --- | --- |
| C | Pascal |
| = | := |
| == | = |
| % | mod |
| / | div |
| ++ | inc |
| int | integer |
| float | real |
| bool | boolean |
| || | или |
| ! | не |
| && | и |
| writeln | printf |
| readln | scanf |
|  |  |
|  |  |
|  |  |
|  |  |

Оглавление

[***Системное программирование*** 1](#_Toc97551592)

[1. Введение 1](#_Toc97551593)

[2. Переменные 4](#_Toc97551594)

[3. Ввод и вывод 5](#_Toc97551595)

[4. Основные размеры блок-схем 7](#_Toc97551596)

[5. Ветвление 9](#_Toc97551597)

[6. Сложные условия 11](#_Toc97551598)

[7. Циклы 16](#_Toc97551599)