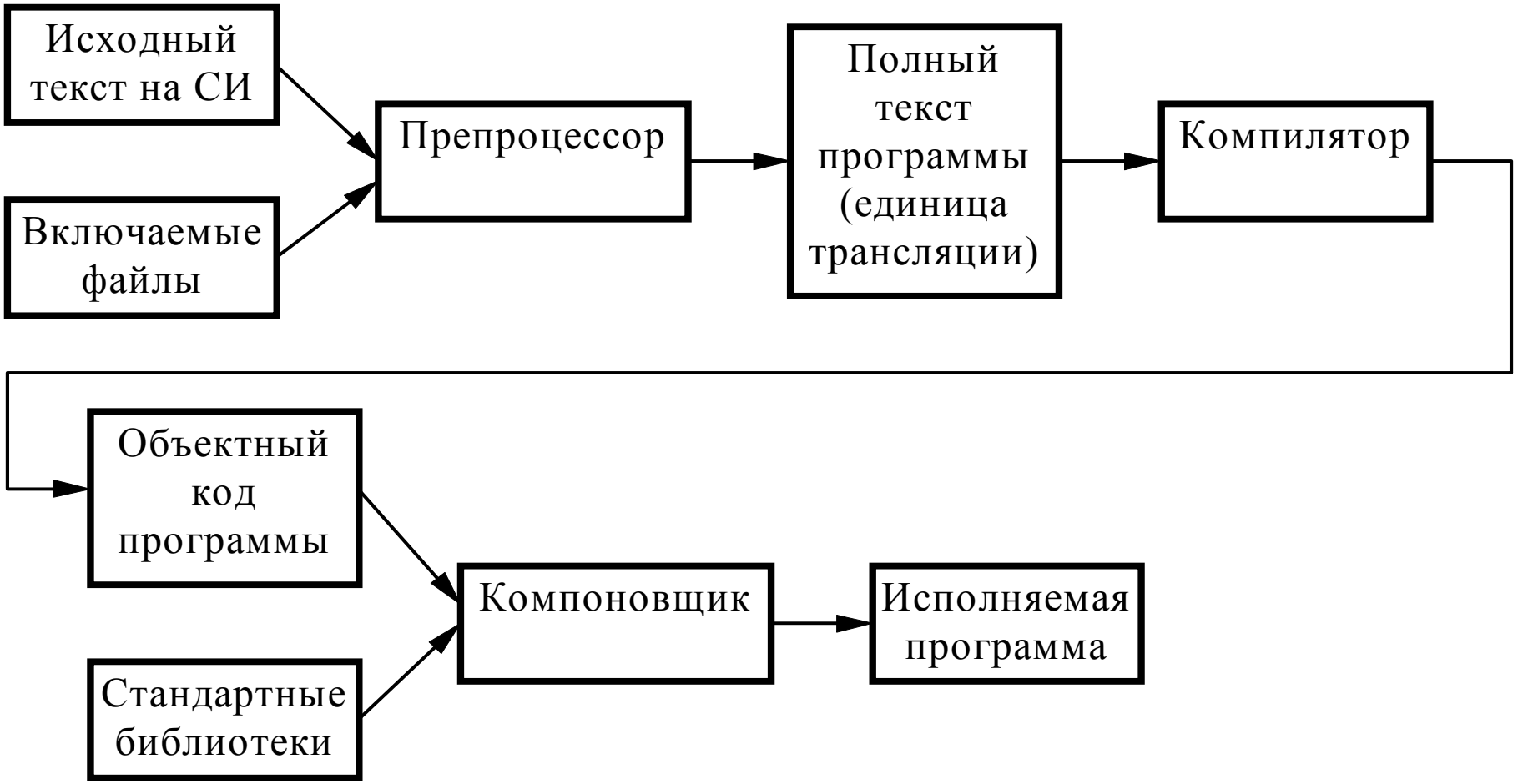
Информатика и информационные технологии

Лекция 2
Язык программирования С

Структура и компоненты простой программы



Функции компонентов

- Препроцессор – вставляет в исходный текст программы содержимое текстового файла типа .h или .hpp вместо директивы #include, описывающее в С интерфейс с определенной библиотекой функций
- Компилятор транслирует исходный код в объектный код, который содержится в файле типа .obj
- Компоновщик объединяет все модули .obj и используемые в программе библиотеки в один загрузочный файл типа .exe

Ключевые понятия любого алгоритмического языка

- Синтаксис языка
  - Как выглядит исходный текст программы
- Семантика языка
  - Чему соответствует в исполняемой программе то, что написано в исходном тексте
- Переменная:
  - Идентификатор (имя переменной)
  - Тип
  - Область видимости (локальные, глобальные переменные)
- Присваивание
  - Значений переменным
- Оператор – синтаксически законченный фрагмент программы, имеющий определенную семантику
- Связывание
  - Имен переменных и функций с их адресом в памяти компьютера
- Процедуры и функции (поименованные части программы, которые можно многократно вызывать в программе)

Структура главного модуля программы

директивы_препроцессора

void main(void)

{

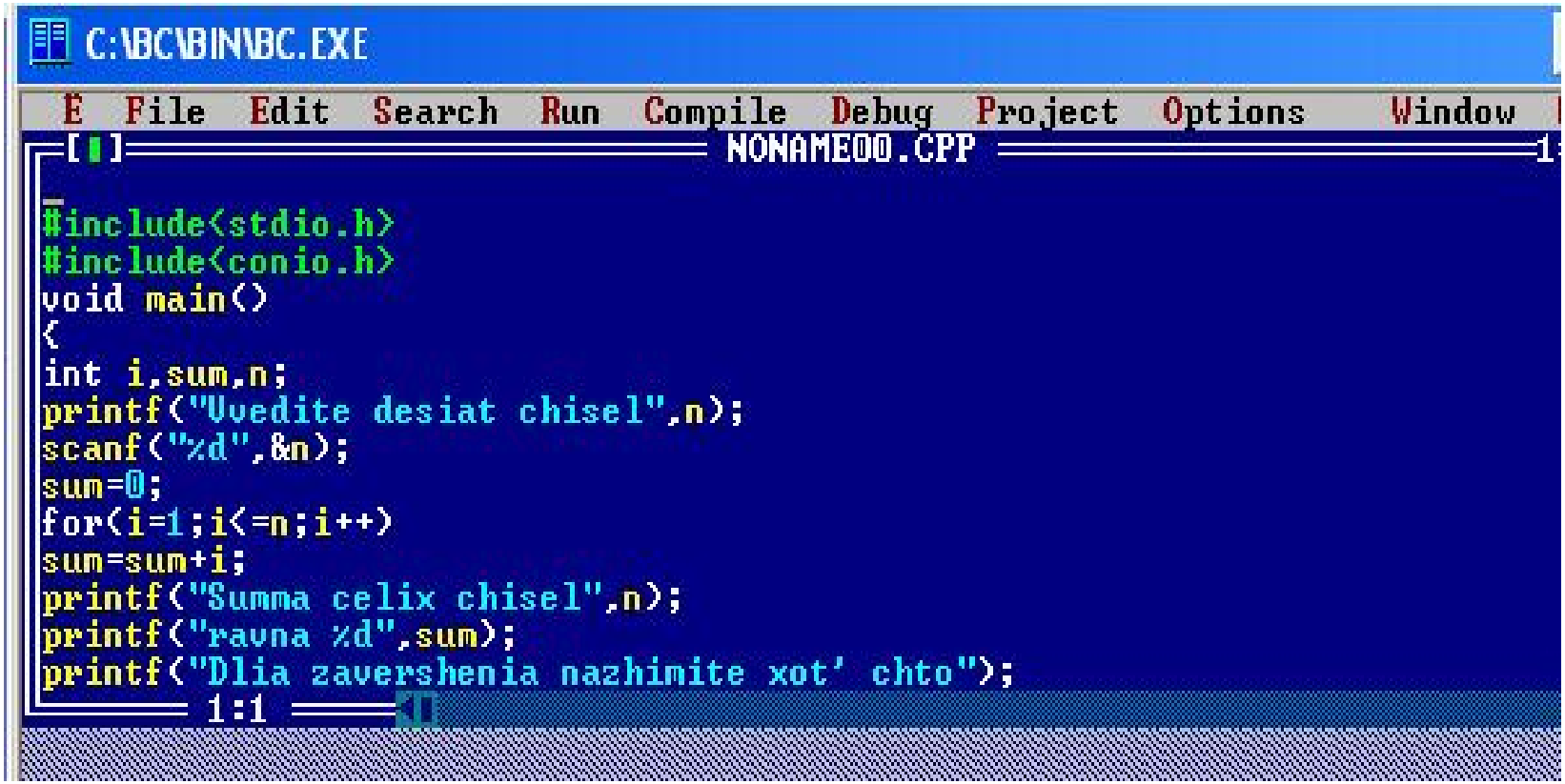
определения_объектов (переменных и констант);

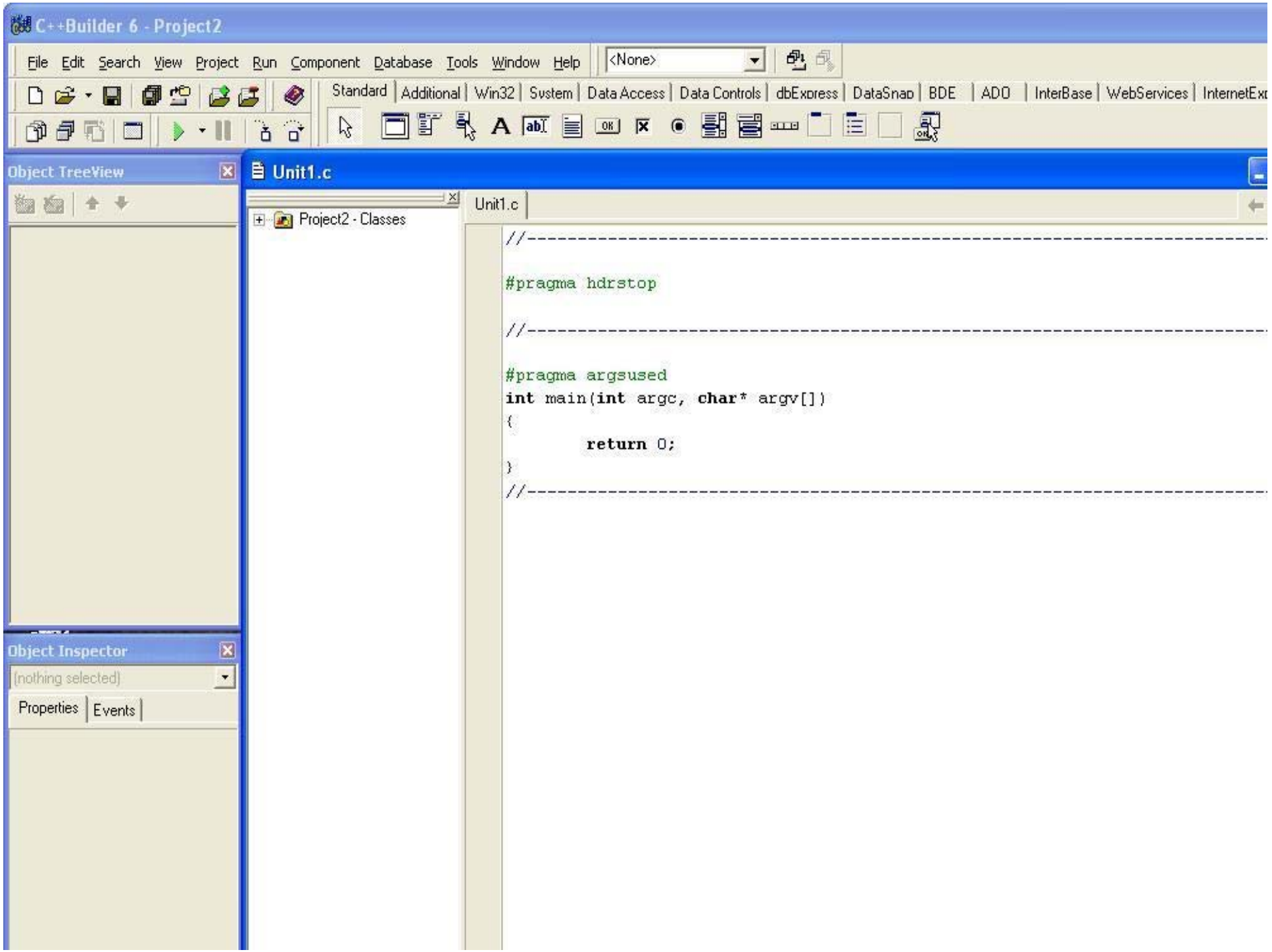
исполняемые_операторы;

}

```
#include <stdio.h>   // включение в текст
                     // файла с функциями
                     // ввода-вывода
/* Пример 1 */
void main(void)
{
   int a, b, sum;
   printf("Введите два числа a и b \n");
   scanf("%d%d", &a, &b);
   sum=a+b;
   printf("a= %d  b= %d  Сумма= %d \n", a,b,sum);
}       // пример вывода: a=10 b=20 Сумма=30
```

Оболочка Borland C++




C++Builder 6 - Project2
File Edit Search View Project Run Component Database Tools Window Help
<None>
Standard Additional Win32 System Data Access Data Controls dbExpress DataSnap BDE ADO InterBase WebServices
Object TreeView
Unit1.c
Project2 - Classes
Unit1.c
//---
#pragma hdrstop
//---
#pragma argsused
int main(int argc, char* argv[])
{
return 0;
}
//---
Object Inspector
(nothing selected)
Properties
Events

Типы данных

Тип	Размер в байтах (битах)	Интервал изменения	
char	1 (8)	от -128	до 127
int	2 (16)	от -32768	до 32767
float	4 (32)	от 3.4E-38	до 3.4E+38
double	8 (64)	от 1.7E-308	до 1.7E+308

void

signed - знаковый

unsigned - беззнаковый

long - длинный

short - короткий

Тип	Размер	Интервал изменения	
char	1 (8)	от -128	до 127
unsigned char	1 (8)	от 0	до 255
signed char	1 (8)	от -128	до 127
int	2 (16)	от -32768	до 32767
unsigned int	2 (16)	от 0	до 65535
signed int	2 (16)	от -32768	до 32767
short int	2 (16)	от -32768	до 32767
unsigned short int	2 (16)	от 0	до 65535
signed short int	2 (16)	от -32768	до 32767
long int	4 (32)	от -2147483648	до 2147483647
signed long int	4 (32)	от -2147483648	до 2147483647
unsigned long int	4 (32)	от 0	до 4294967295
float	4 (32)	от 3.4E-38	до 3.4E+38
double	8 (64)	от 1.7E-308	до 1.7E+308
long double	10 (80)	от 3.4E-4932	до 3.4E+4932

Объявление переменных

тип <список_переменных>;

Пример:

```
int x,y,z;
float rad;
unsigned char ch;
long double intgr;
```

тип имя_переменной=начальное_значение;

```
float pi=3.1415, cc=1.23;
```

```
char a=130;        // реальное значение a=-126
```

Константы

1. Символы, или символьные константы

‘A’, ’a’, ’+’, ’;’

2. Целые константы

44 684 0 1024

3. Вещественные константы

44. 3.14159 44e0 .314 0.0

4. Строковые константы

H	E	L	L	O	\0

Определение констант

const *тип имя_константы=значение_константы;*

```
const double E=2.718282;
const long M=99999999;
const F=765;
```

#define *имя_константы значение_константы*

```
#define NEXT ‘Z’
```

```
#include <limits.h>
#include <values.h>
```

Операции языка СИ

Арифметические

-	Вычитание и унарный минус		A-B	-A
+	Сложение		A+B	
*	Умножение		A*B	
/	Деление	float a; a=1/2;	int a; a=11/3;	
%	деление по модулю	a=6%2	a=7%3	
++	увеличение на единицу	x=x++	x=++x	
--	уменьшение на единицу	x=x--	x=--x	

x++

++x

Пример:

```
int k=3, a=10, x1, x2;
x1=++k;    // 4
x2=a++;    //  10
x1=k;      //   4
x2=a;      //   11
k++;       //    5
```

#include <math.h>

abs(X) вычисление абсолютного значения
cos(X)
exp(X)
sin(X)
sqrt(X) корень
log(X) натуральный логарифм
pow(x,y) x^y

Пример: $\dfrac{a\sqrt{\sin(bc+a)}-e^{-ac}}{2b+d}$

(a*sqrt(sin(b*c+a))-exp(-a*c))/(2*b+d)

Y=X*Pi/180

$\log_a(x)=\log(x)/\log(a)$

Операции отношения и логические операции

<	меньше	&&	и
< =	меньше или равно	||	или
>	больше	!	не
> =	больше или равно		
= =	равно		
!=	не равно		

Операция присваивания
знак “=“ вместо “:=” в Паскале

```
x=2*x;

a=b=c=x*y;    // присваивание нескольким
              // переменным одного значения

m-=20;      m=m-20;
m+=20;      m=m+20;
m*=20;      m=m*20;
m/=10;      m=m/10;
m%=10;      m=m%10;
```

Ввод-вывод

#include <stdio.h>

Функция форматированного вывода на экран:

printf(форматная_строка, список_аргументов);

Управляющие символы:

‘\n’ – перевод строки;

‘\t’ – горизонтальная табуляция;

‘\0’ – нулевой символ;

‘\a’ – сигнал-звонок.

Спецификации преобразования:

%флаги ширина_поля.точность модификатор спецификатор %спецификатор

Спецификаторы:

%d – целое число;
%u – целое число без знака;
%p – число в формате указателя;
%f – число с плавающей точкой;
%e – в экспоненциальной форме;
%c – символ;
%s – строка;
%x – целое в шестнадцатеричном формате.

Модификаторы
h – shot int;
l – long int;
L – long double.

- Пример:

```
int k=-83;
float a=22.789, b=11.34;
printf("\nk=%d\ta=%.2f\nb=%e",k,a,b);
```

- k=-83 a=22.79
- b=1.134000e+01

#include <stdio.h>

Функция форматированного ввода:

scanf(форматная_строка, список аргументов);

форматная_строка:

%*ширина_поля модификатор спецификатор

%спецификатор

```
int a;
float b, c;
scanf ("%d%f%f", &a, &b, &c);
```

Потоковый ввод-вывод

```
#include <iostream.h>
```

Формат вывода на экран:

```
cout << выражение;
```

Формат ввода:

```
cin >> переменная;
```

```
int a;
cout << "Enter: ";
cin >> a;
cout << "a=" << a << endl;
```

Управляющие операторы

Оператор выражение

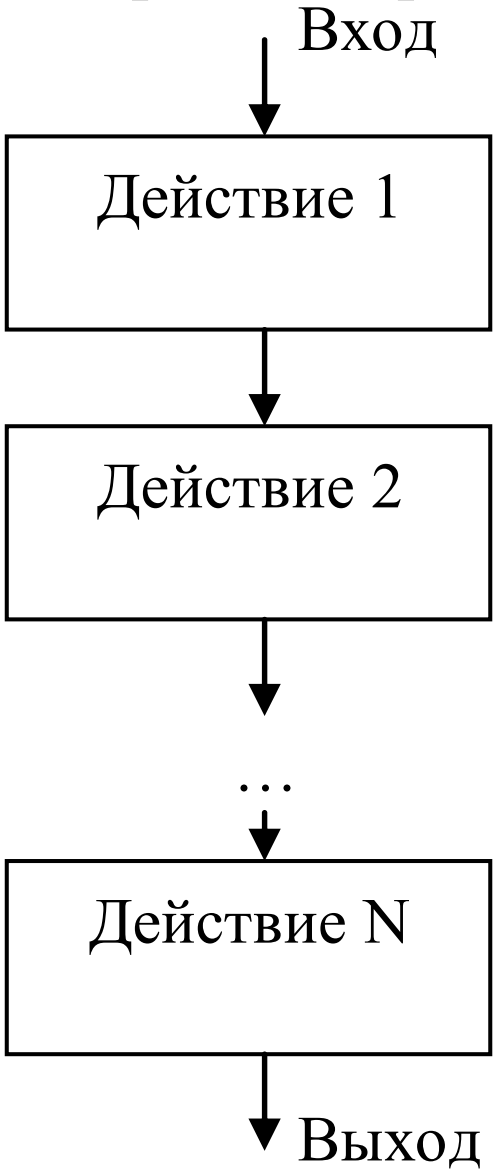
```
++i;
a=cos(b * 5);
```

Пустой оператор

```
;
```

Составной оператор

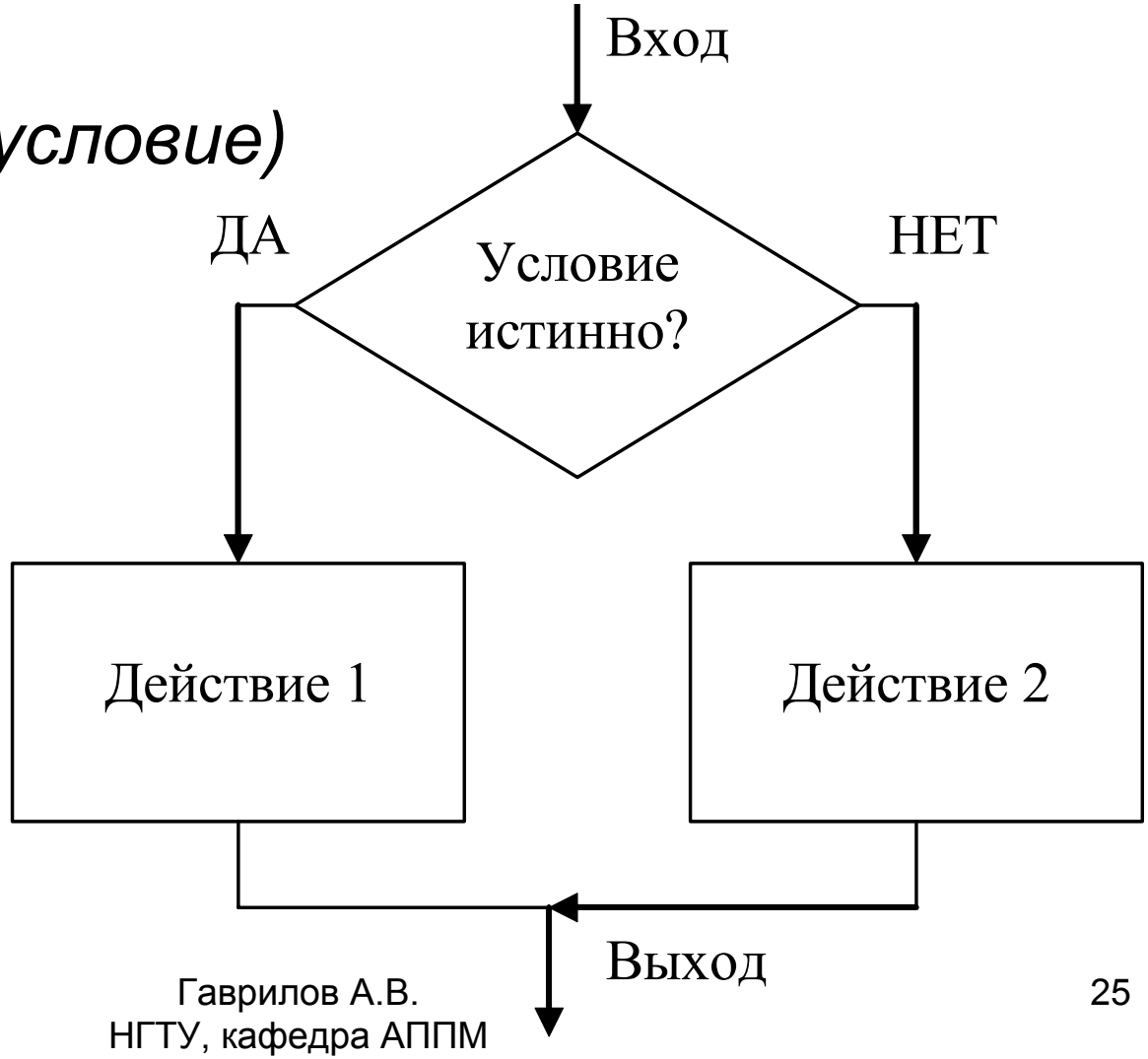
```
{  оператор1;
   оператор2;
    …;
   операторN;
}
```



Условные операторы

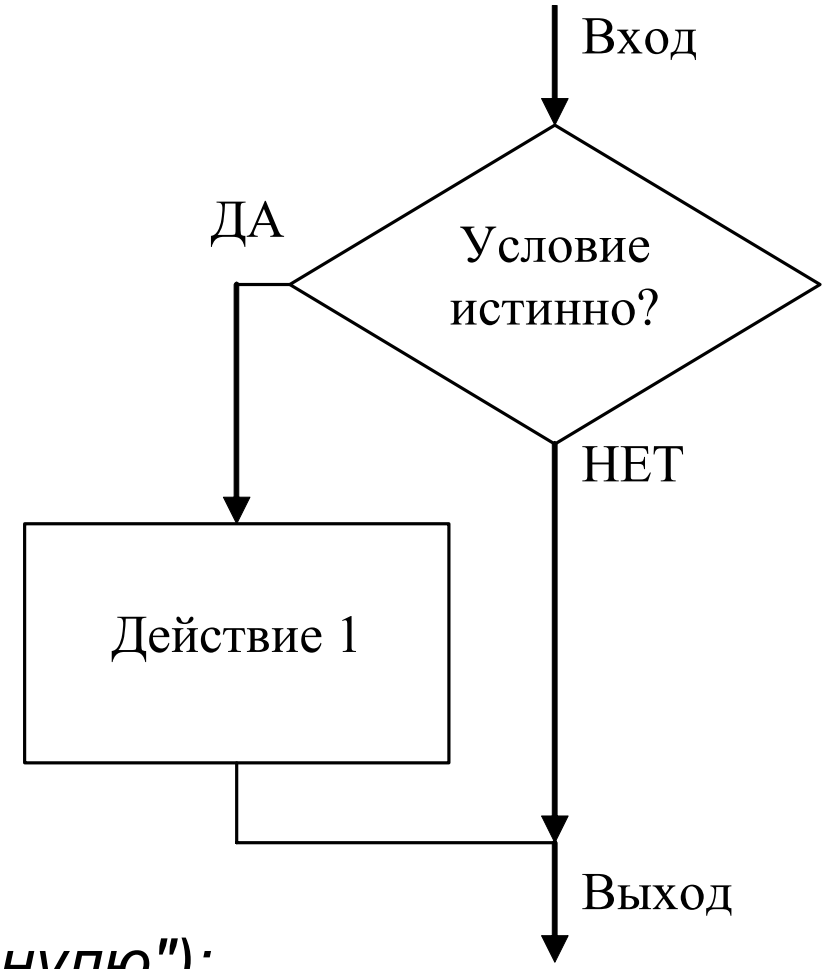
Условный оператор if

if *(выражение_условие)*
оператор_1;
else
оператор_2;



if (выражение_условие***)*** оператор;

if (выражение_условие)
{ оператор_11;
оператор_12;
}
else
{ оператор_21;
оператор_22;
}



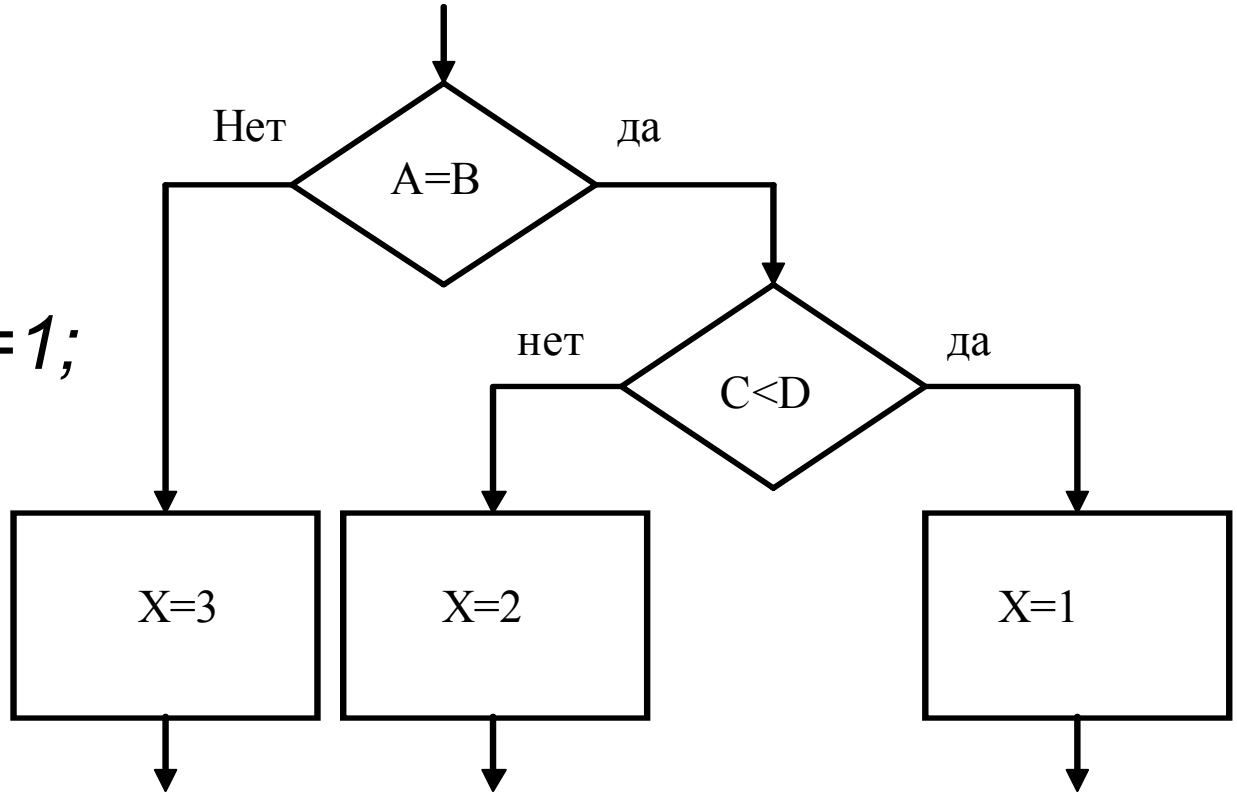
Пример1:

```
if (x) printf("число не равно нулю");
else printf("число равно нулю");
```

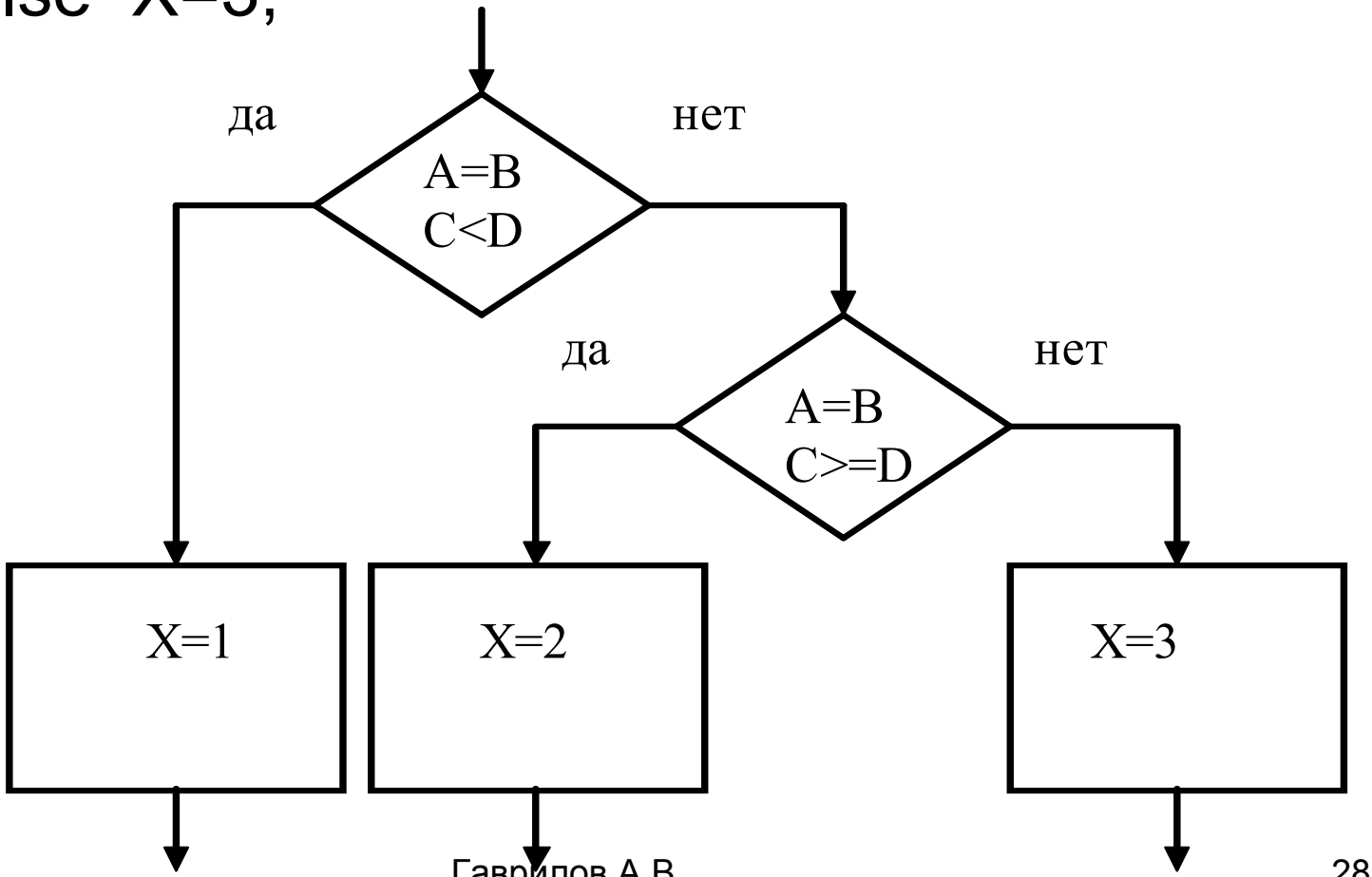
Пример2:

$$X = \begin{cases} 1 \text{ при } A = B \text{ и } C < D \\ 2 \text{ при } A = B \text{ и } C >= D \\ 3 \text{ при } A <> B \end{cases}$$

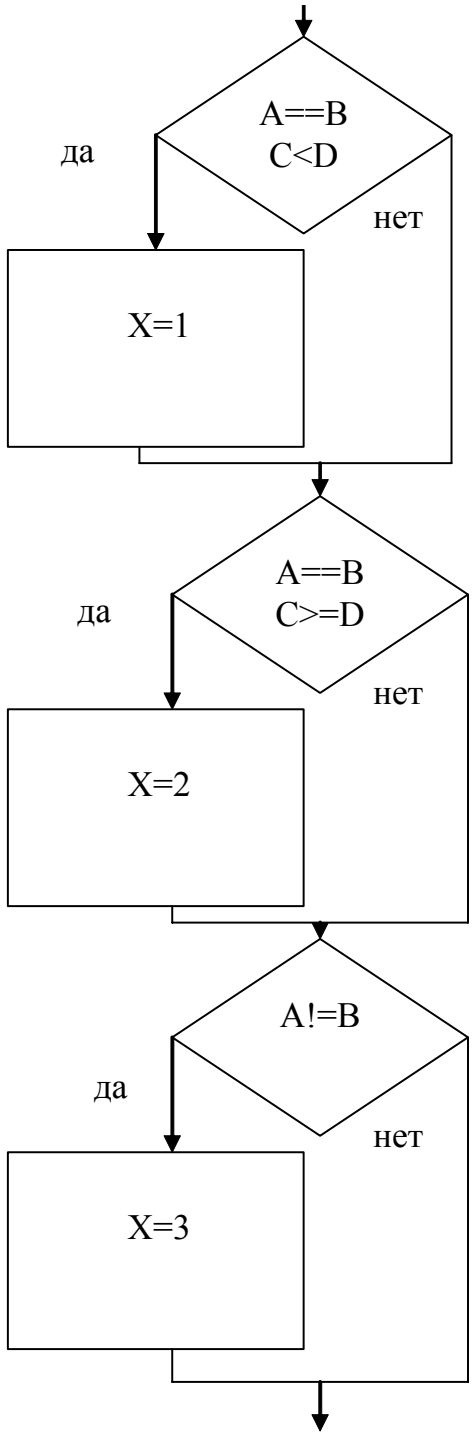
```
if (A==B)
    if (C<D)  X=1;
     else X=2;
else X=3;
```



```
if (A==B && C<D) X=1;
else if (A==B && C>=D) X=2;
        else  X=3;
```



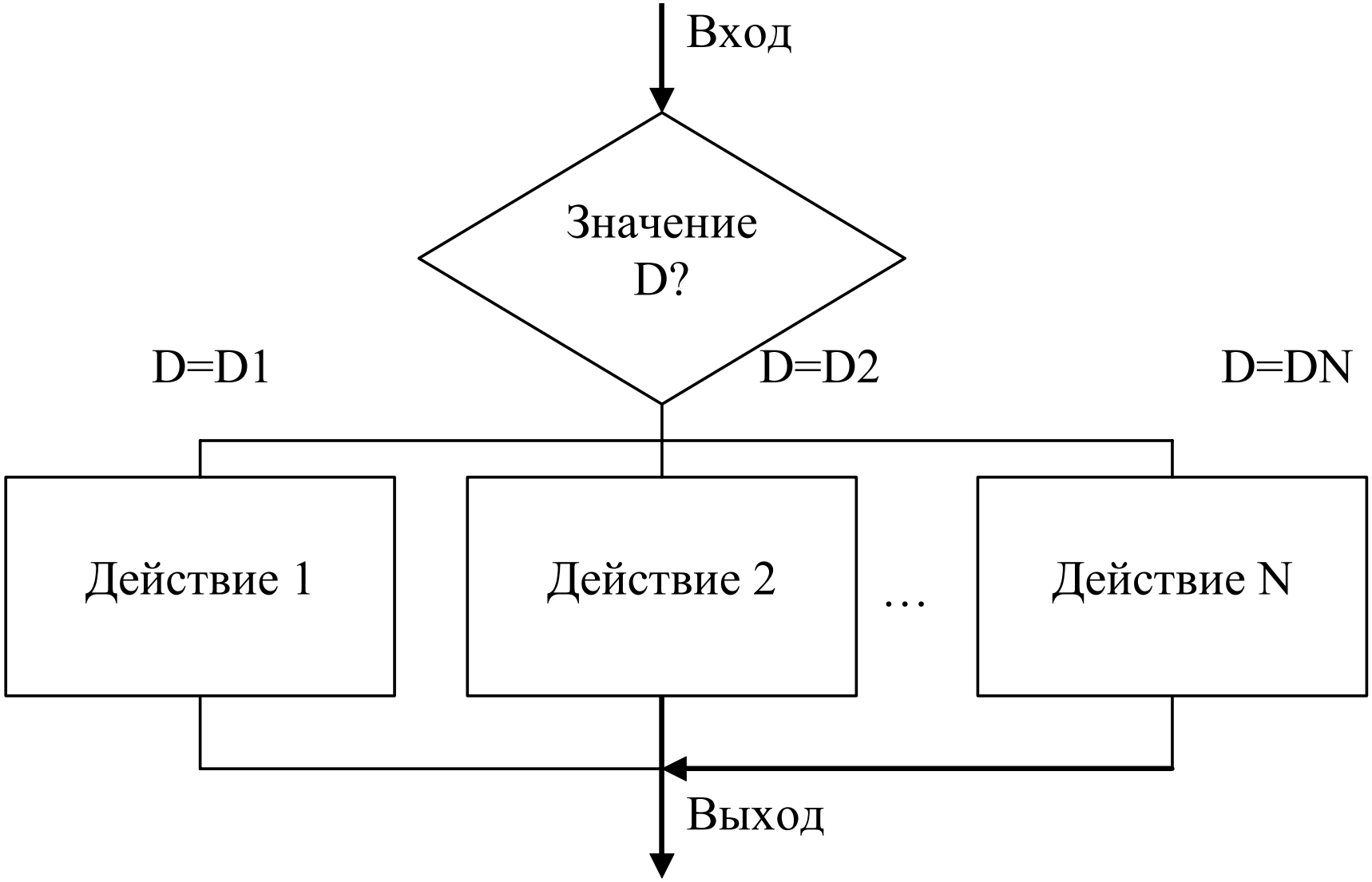
```
if (A==B && C<D) X=1;
if (A==B && C>=D) X=2;
If (A!=B)  X=3;
```



$$y=\begin{cases} x^2, 0<x<10 \\ x^3, 10\le x<100 \\ x, x\ge 100 \\ 0, x\le 0 \end{cases}$$

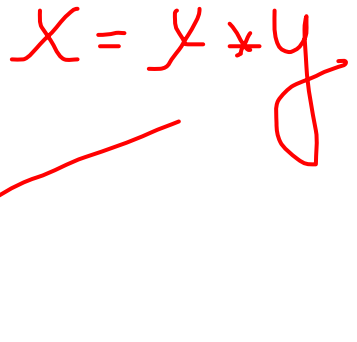
```
if (x>0&&x<10) y=x*x;
if (x>=10&&x<100) y=x*x*x;
if (x>=100) y=x;
if (x<=0) y=0;
```

Оператор ***switch***



```
switch (выражение)
{
case const_1:
   последовательность операторов;
   break;
case const_2:
   последовательность операторов;
   break;
case const_3:
   последовательность операторов;
   break;
default операторы;
    }
```

Пример:



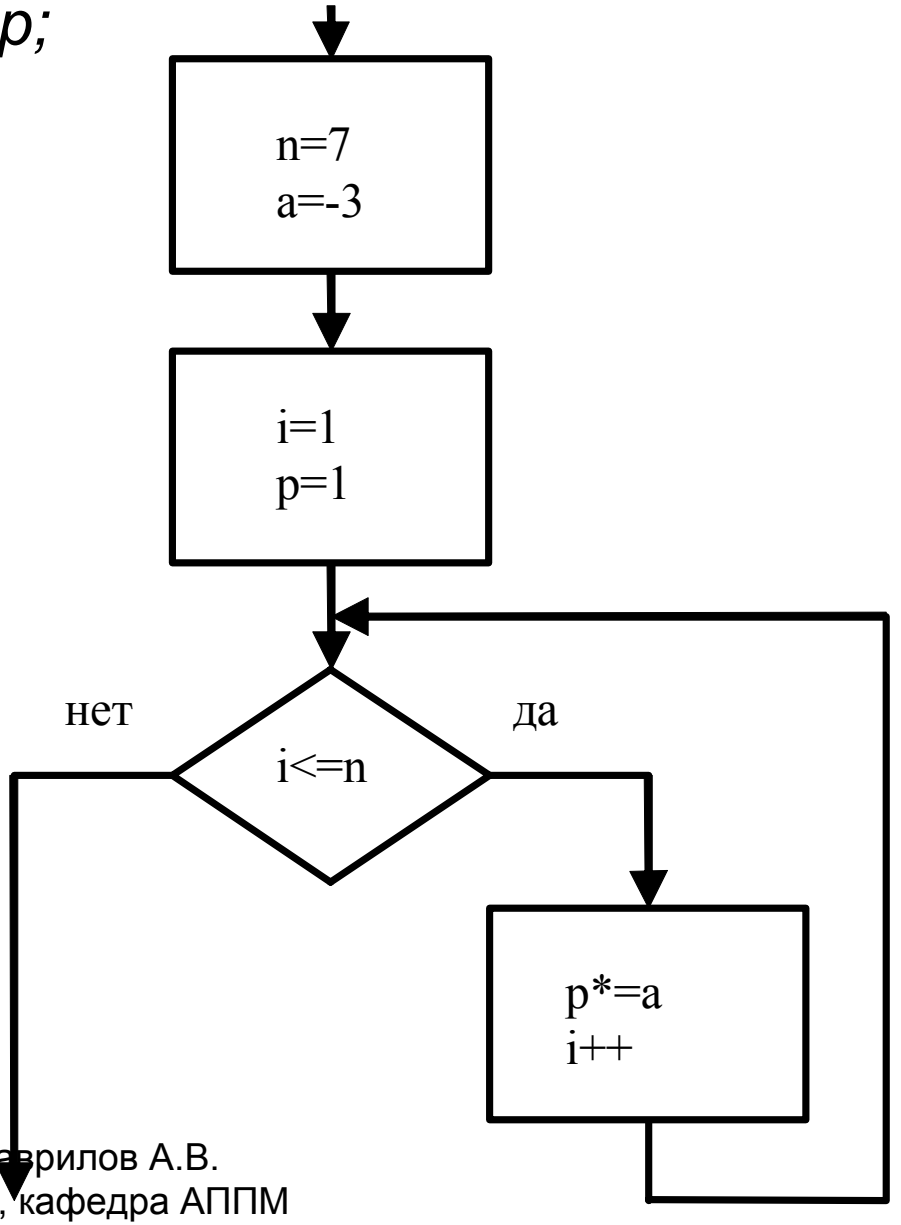

```
switch (operand) {
   case MULTIPLY:   x *= y; break;
   case DIVIDE:     x /= y; break;
   case ADD:        x += y; break;
   case SUBTRACT:   x -= y; break;
   case INCREMENT: x++;    break;
   default:         printf("Bug!\n");
 }
```

Операторы цикла

while *(условие) оператор;*

<u>Пример</u>: Вычислить A^N:

```
…
n=7; a=-3;
i=1; p=1;
while (i<=n)
    {
    p*=a;
    i++;
    }
…
```



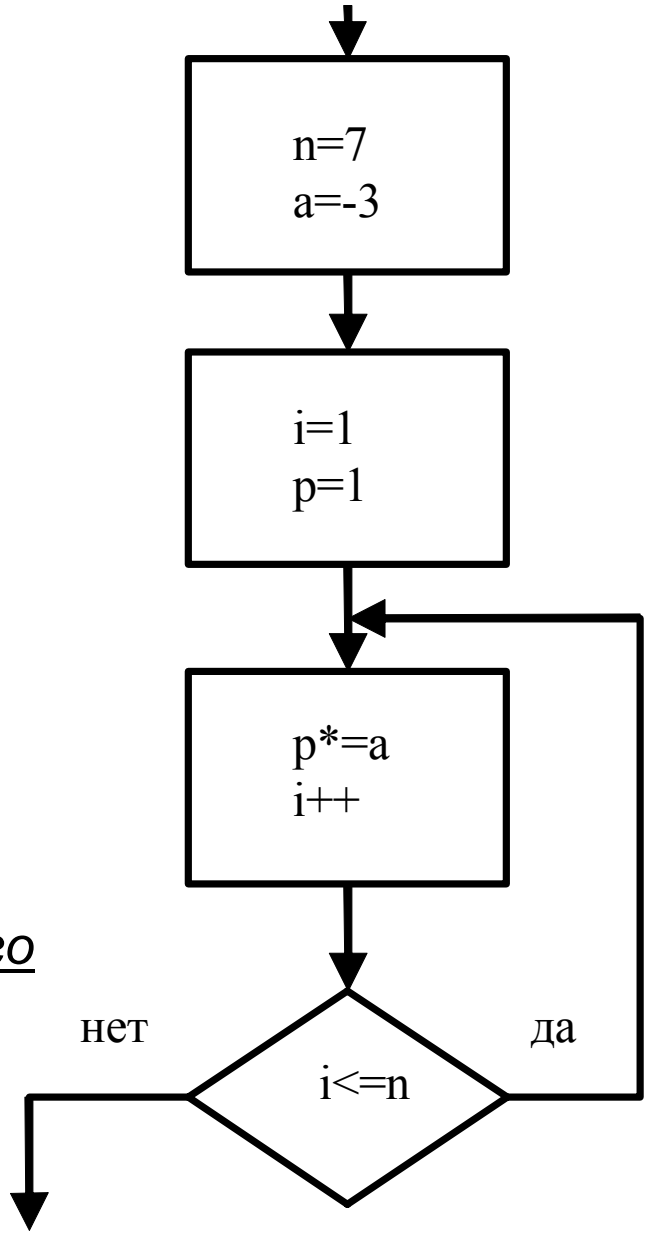
do *оператор;*
while *(условие);*

Пример 1: Вычислить A^N*:*

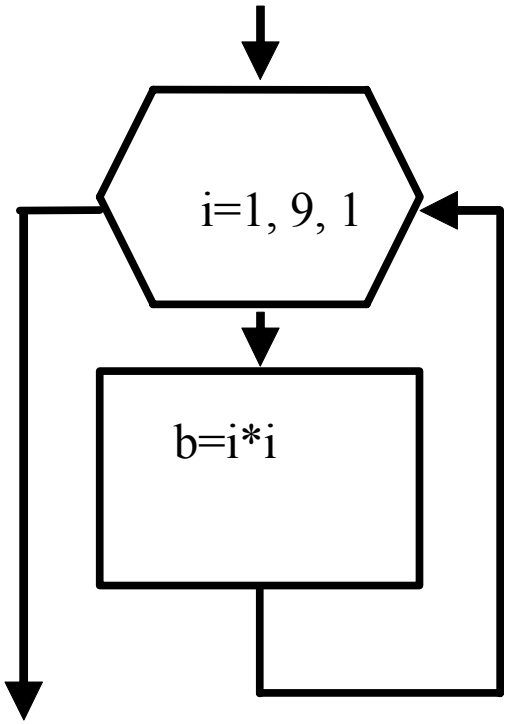
```
...
n=7;a=-3;
i=1; p=1;
do {
    p*=a;
    i++;
    } while (i<=n);
...
```

Пример 2 (ввод целого положительного числа не больше 4):

```
do
scanf("%d",&i);
while ((i<=4)||(i>=0)) ;
```



for (выражение1; выражение2; выражение3) оператор;
Где:
Выражение1 – начальные значения;
Выражение2 – условие продолжения цикла
Выражение3 – изменение переменных
в каждой итерации



Пример1: вывод квадратов чисел от 1 до 9

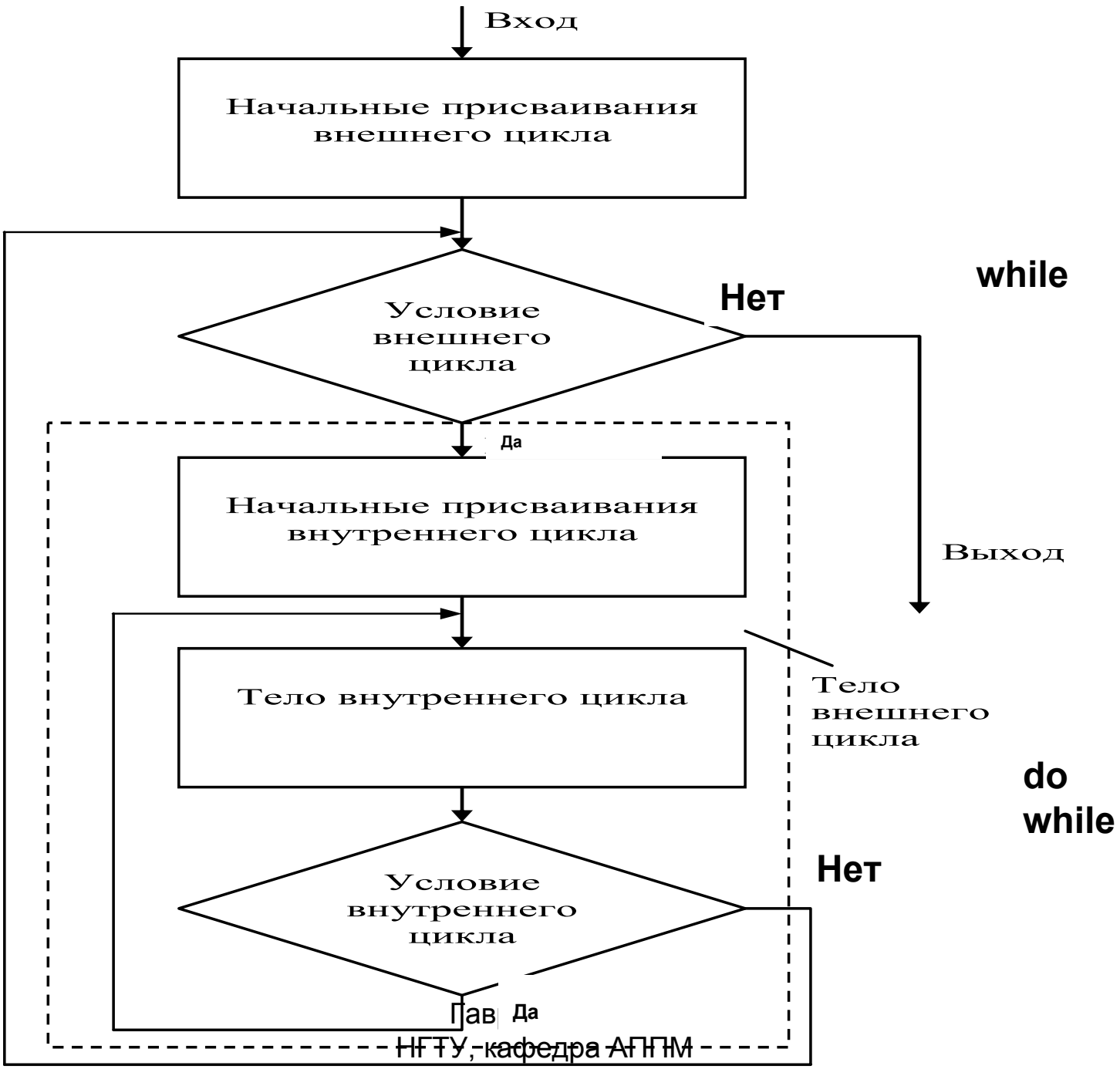
```
void main(void)
    { int i,b;
      for (i=1; i<10; i++)
        {b=i*i;printf("%d ",b);}
    }
```

Пример 2:

```
for (char ch='0'; ch!='a';) scanf("%c", &ch);  // ждет ввода 'a'
```

Пример 3:

```
for (int i=9; i>=0; i--) printf("%d\n", i); // вывод цифр от 9 до 0
```


Вход
Начальные присваивания внешнего цикла
Условие внешнего цикла
Нет
Да
Начальные присваивания внутреннего цикла
Тело внутреннего цикла
Условие внутреннего цикла
Нет
Да
Выход
Тело внешнего цикла
while
do while

Операторы *break* и *continue* в циклах

Пример:

```
for (i=1; i<100; i++)
{
...
if (i==50) break;
}
printf("i =%d“, i);
```

Пример:

```
for (i=1; i<100; i++)
{
if (i%7) continue;
  printf("%d\n", i) ;
}
```

Оператор *goto*

```
goto имя-метки;
        ...
имя-метки: оператор;
```

Пример:

```
for (i=1; i<100; i++)
{
...
  if (i==50) goto a;
}
...
a: printf("i = 50");
```

Операция запятая

Пример:

```
int k;
printf("\n%d", (k=3,2*k));
printf("\nk=%d", k);
```

Условная трехместная операция

выражение_1 ? выражение_2 : выражение_3;

Пример:

Поиск максимума `max=(x>y) ? x : y;`

Поиск абс. Величины `abs=(x>0) ? x : -x;`

Наибольшее из чисел заменить на 1

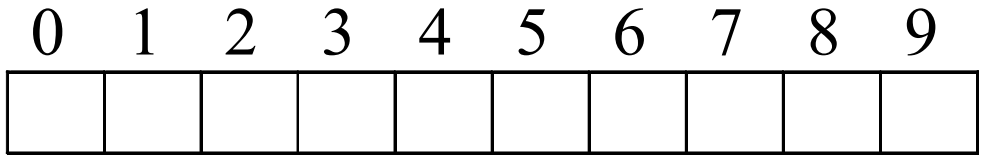
`(x>y) ? x : y=1;`

Массивы

тип имя_массива[размер1][размер2]...[размерN];

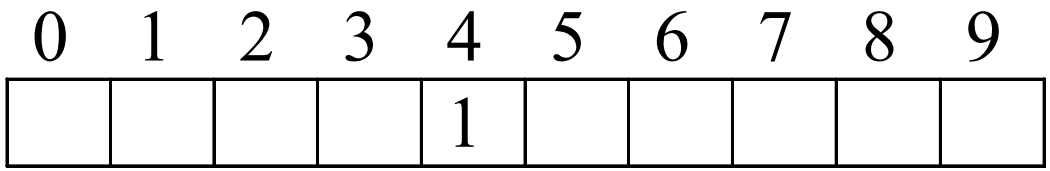
Одномерные массивы

float arr[10];
arr[0] … arr[9]



1. double d[ ]={1.0, 2.0, 3.0, 4.0, 5.0};
2. int M[8]={8, 4, 2};

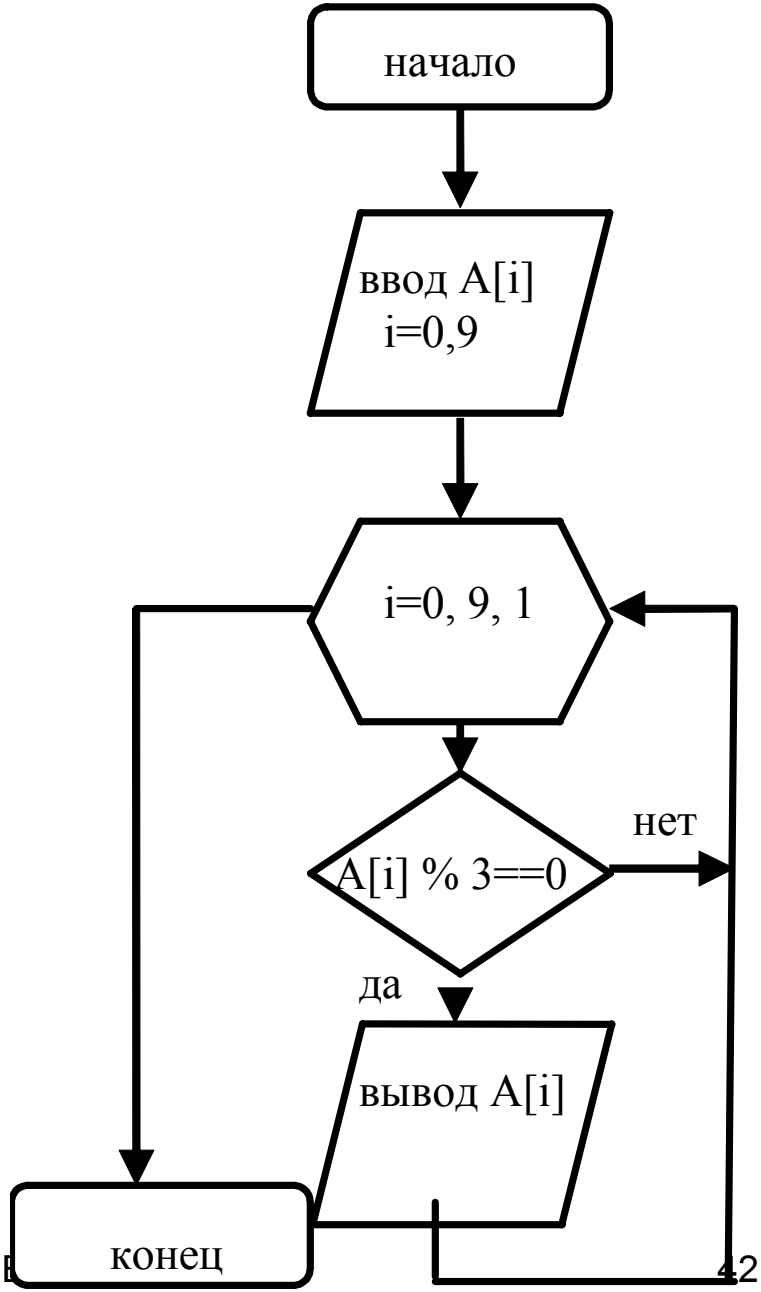
int A[10];
A[4]=1;



for (i=0; i<10;i++) scanf(“%d”,&A[i]); // ввод массива целых чисел
for (i=0; i<10;i++) printf(“%d ”,A[i]); // вывод массива целых чисел

Пример:

```
#include <stdio.h>
void main(void)
{
int i;
int mas[10];
printf("\nEnter array:\n");
for(i=0;i<10;i++)
    scanf("%d",&mas[i]);
for(i=0;i<10;i++)
    if (!(mas[i]%3))
    printf("%d\t",mas[i]);
}
```



Некоторые приемы алгоритмизации

Обмен значениями между двумя переменными A и B

C = A

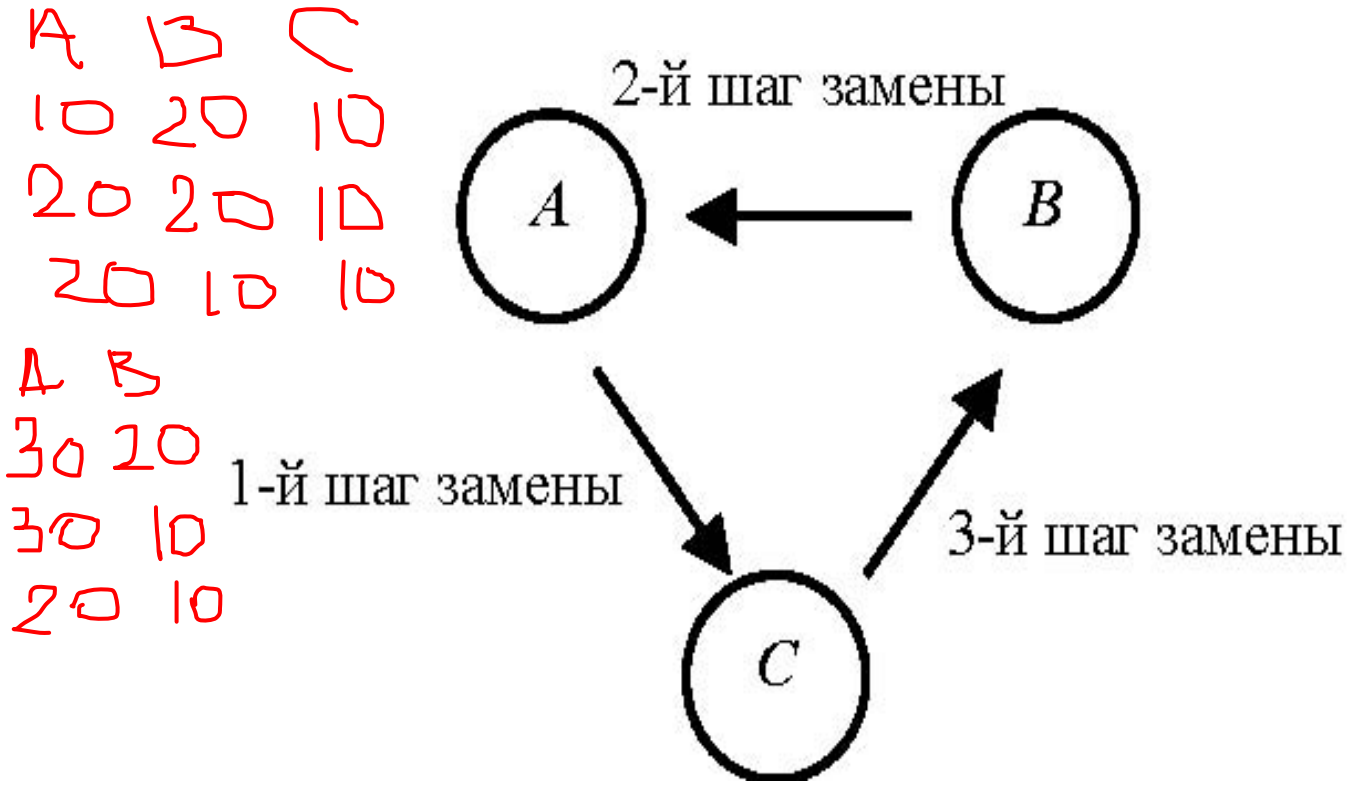
A = B

B = C

A=A+B

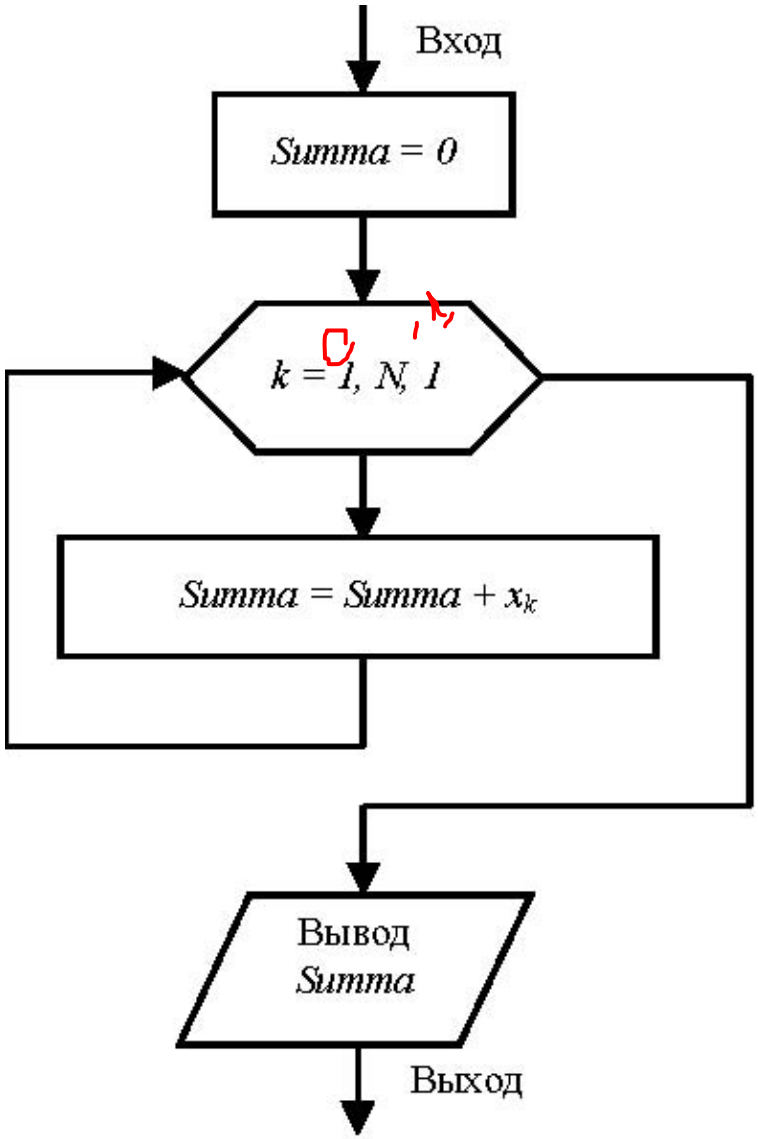
B=A-B

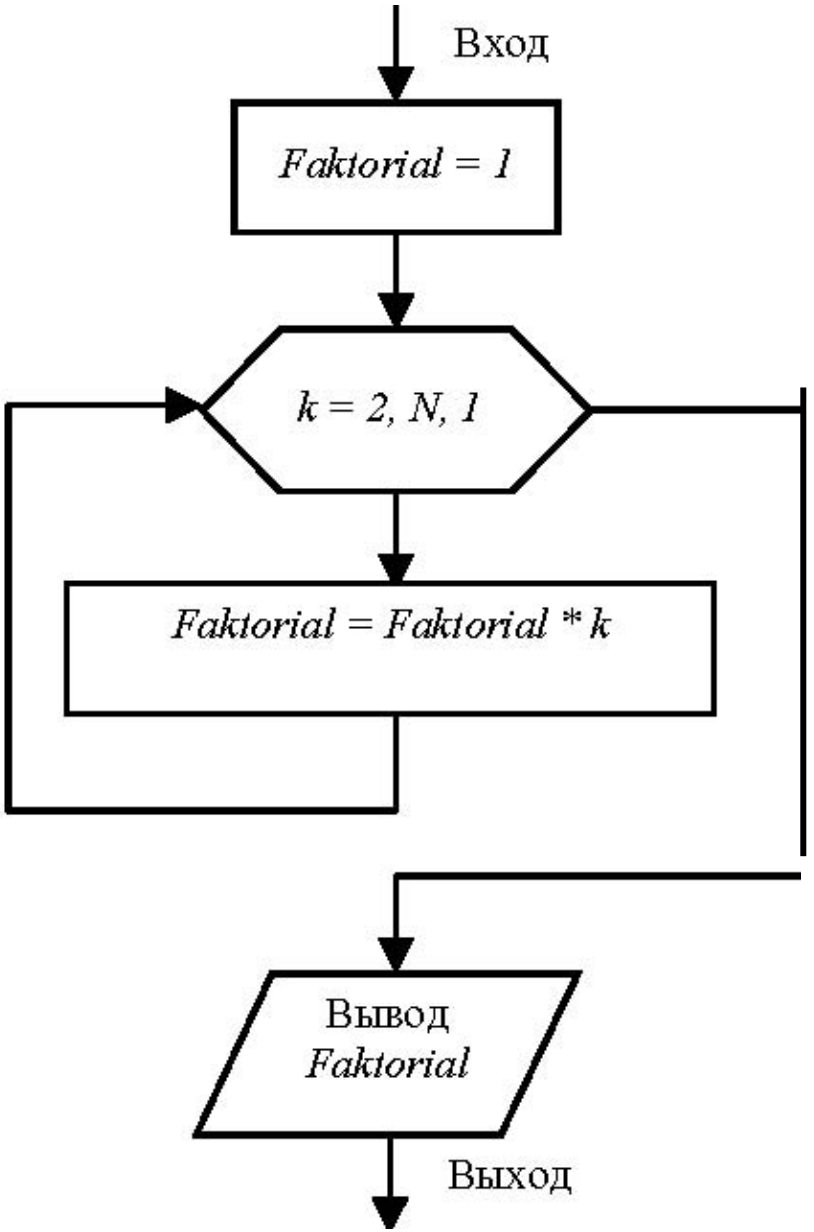
A=A-B



Вычисление произведения (суммы) чисел

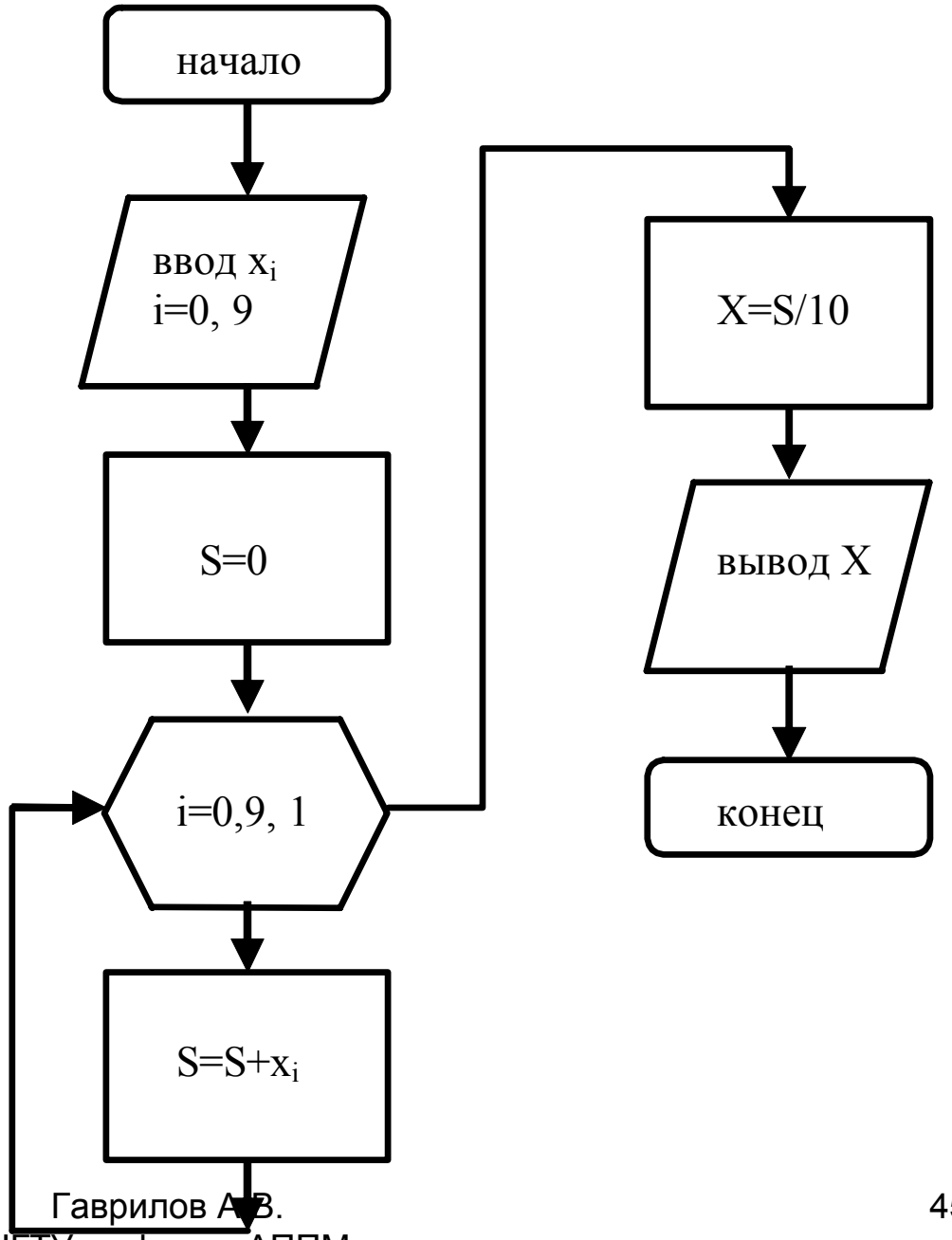
(основаны на накоплении результата в переменной)





$$X = \frac{x_1 + x_2 + \ldots + x_{10}}{10} =$$

$$= \frac{1}{10} \sum_{i=1}^{10} x_i$$



Многомерные массивы и операции с ними

```
int a[2][3];
[1      2      5]
[6      7      2]
```

1	2	3	4	5
6	7	8	9	10
11				

```
double A[3][2]={{10,20}, {30,40}, {50,60}};
double A[3][2]={10,20,30,40,50,60};

int A[3][5]={1,2,3,4,5,6,7,8,9,10,11};
int A[3][5]={{1,2,3},{4,5,6,7,8},{9,10,11}};
int arr[][3]={1, 2, 3,
              4, 5, 6,
              7, 8, 9};
```

1	2	3		
4	5	6	7	8
9	10	11		

```
int arr[2][3];
for (i=0; i<2;i++)
for (j=0; j<3;j++) scanf(“%d”,&arr[i][j]);

for (i=0; i<2;i++) {
for (j=0; j<3;j++)
printf(“%d  ”,arr[i][j]);
    printf(”\n”);
    }
```

1 2 5

6 7 2

```
int a[2][3];
int at[3][2];
int i,j;
printf("Enter array\n");
for (i=0;i<2;i++)
for (j=0;j<3;j++) {
scanf("%d",&a[i][j]);
at[j][i]=a[i][j];
   }
printf("Транспонированный массив\n");
for (i=0;i<3;i++) {
for (j=0;j<2;j++) printf("%d ",at[i][j]);
   printf("\n");}
```

A[4][5], B[4][5]

…

```
for (i=0;i<4;i++)
for (j=0;j<5;j++) C[i][j]=a[i][j]+b[i][j];
```

…

A[M][N] B[N][L]
C[M][L]

```
const M=3, N=4, L=5;
int a[M][N];
int b[N][L];
int c[M][L];
```

…

```
for(i=0;i<L;i++)
for(k=0;k<M;k++)
for(j=0;j<N;j++) c[k][i]=c[k][i]+a[k][j]*b[j][i];
```

...

Символьная информация и строки

0 … 255 -128 … 127

```
char a, z, symb;
```

Пример:

```
unsigned char ch;
scanf("%c",&ch);
printf("вы ввели символ - %c",ch);

char x;
x=getchar();
putchar(x);
```

Пример: ввод в массив mas символов до точки с пропуском пробелов

```
#include <stdio.h>
void main(void)
{char ch, mas[50];
int k;
for (k=0;(ch=getchar())!='.';)
{  mas[k]=ch;
if (ch!=' ') k++;
}
}
```

Кодировка символов в ASCII

0..9 - 48..57

A..Z - 65..90

a..z - 97..122

А..Я - 128..159

а..п - 160..175

р..я - 224..239

Пример 1

```
#include <stdio.h>
void main(void)
{
char z;
for (z='0';z<='9';z++) printf("%c\t%d\n",z,z);
}
```

Пример 2

```
#include <stdio.h>
void main(void)
{
char z;
for (z='A';z<='Z';z++) printf("%c",z);
}
```

```
#include <conio.h>
X=getch();
putch(X);
while((ch=getch())!=‘z’);
```

Строки

- одномерные массивы символов,
заканчивающиеся символом с кодом ноль (‘\0’)

```
char str[20];
str[0]='H'; str[1]='e'; str[2]=str[3]='l';
str[4]='o'; str[5]='\0';
char str[]={'H','e','l','l','o','\0'};
char* str1="Hello";
char str[]="Привет";
char str[80]= "Привет";
```

Специальные символьные константы

\a	Сигнал	**\t**	Горизонтальная табуляция
\f	Подача бумаги	\v	Вертикальная табуляция
\n	Новая строка	\’	Одинарная кавычка
\r	Возврат каретки	\xN	Шестнадцатеричная или восьмеричная константа
\?	Знак вопроса	\N	
\”	Двойная кавычка	\b	Удаление предыдущего символа
****	Обратный слэш		

Ввод-вывод строк

1. При помощи функций printf() и scanf()

Пример:

```
char c[80];
scanf(“%s”,&c);
printf(“%s”,c);
```

2. Специальные функции для ввода-вывода строк

gets(); **puts();**

Пример:

```
char ch[80];
printf("\nвведите строку\n ");
gets(ch);
puts(ch);
```

Функции для работы со строками

```
                                        #include <string.h>
strcpy(s1,s2);        // копирование строки
strcpy(s1,"Hello ");
char *s=strcat(s1,s2);  // результат –
   конкатенация строк
int n=strcmp(s1,s2); // сравнение строк
int n=strlen(s); // длина строки
int x=atoi(s);  // преобразование строки в число
```

Переименование типов - typedef

typedef тип новое_имя_типа;

typedef тип новое_имя_типа[размер_массива];

Пример:

```
typedef char SYMBOL;
typedef char STRING[80];
typedef int MASS[10];

SYMBOL ch;
STRING str;
MASS a;
```