Основы языка Си/Си++

Тема 19

ЭЛЕМЕНТЫ ЯЗЫКА СИ++

- ***Алфавит***
  - латинские буквы (**A-Z, a-z**)
  - цифры **0-9**
  - специальные символы **(, % ! .** и другие

Замечание 1 : в комментариях, строках и символьных константах могут использоваться и другие символы (например, русские буквы).

Замечание 2: комбинации некоторых символов, не разделен-ных пробелами, интерпретируются как один значимый символ.

++, --, += и т.д.

- ***Ограничители комментариев***

/* Это комментарий языка Си */

// Это строчный комментарий Си++

ЭЛЕМЕНТЫ ЯЗЫКА СИ++

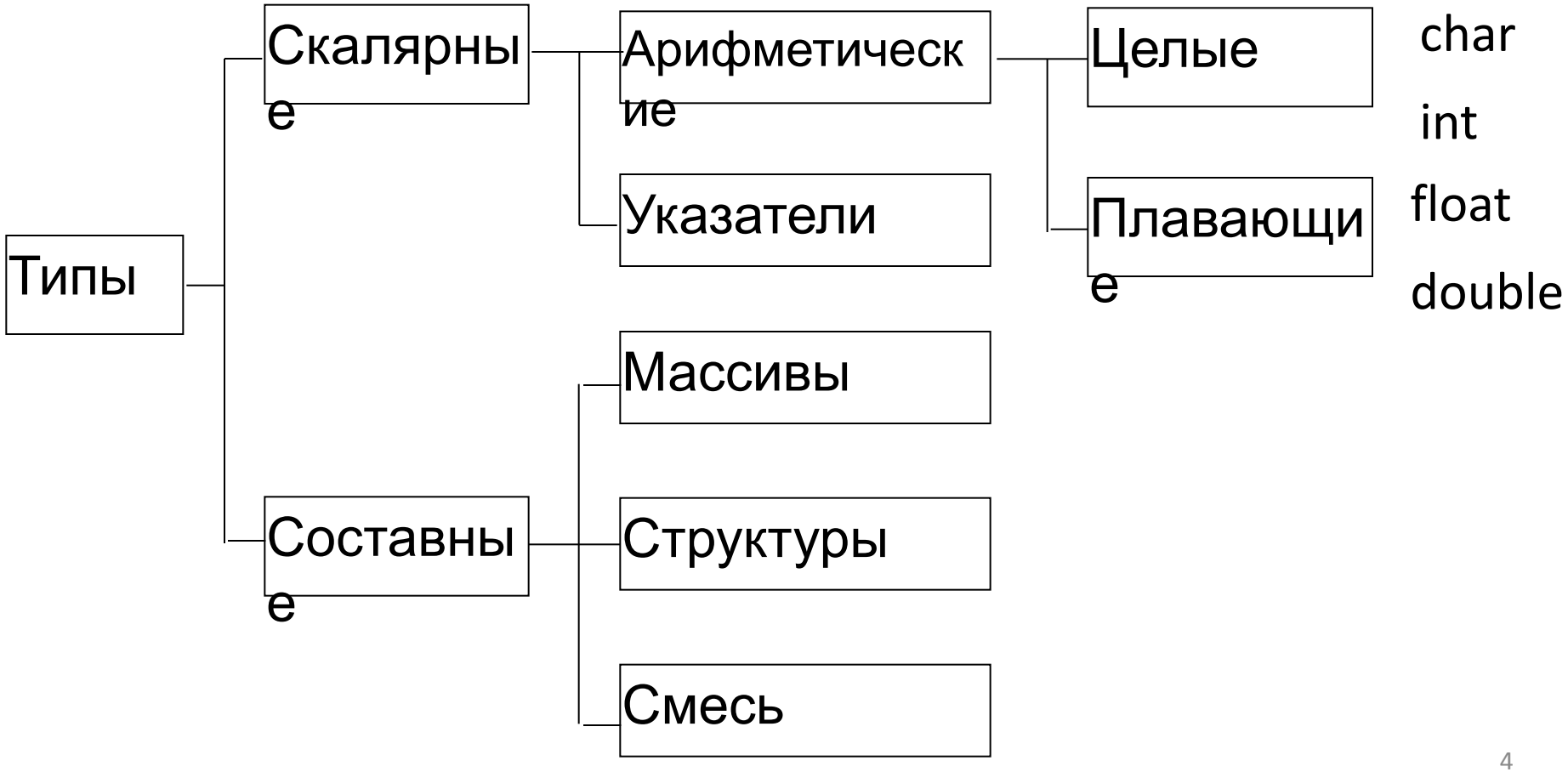
Из символов алфавита формируются **лексемы** – единицы текста программы, которые при компиляции воспринимаются как единое целое и не могут быть разделены на более мелкие элементы

Виды лексем:

- идентификаторы (последовательность букв, цифр, символов подчеркивания, начинающаяся с буквы или символа подчеркивания. Прописные и строчные буквы различаются)
- служебные (ключевые)слова (это идентификаторы, назначение которых однозначно определено в языке)
- константы
- знаки операций

ТИПЫ ДАННЫХ ЯЗЫКА СИ++

В языке СИ++ имеется 4 базовых арифметических типа и 2 модификатора (знака и длины)



АРИФМЕТИЧЕСКИЕ ТИПЫ ДАННЫХ ЯЗЫКА СИ++

Тип данных	Размер (байт)	Диапазон значений	Эквивалентные названия типа
char	1	–128 ... +127	signed char
int	2/4	зависит от системы	signed, signed int
unsigned char	1	0...255	нет
unsigned int	2/4	зависит от системы	unsigned
short int	2	–32768 ... 32767	short, signed short int
unsigned short	2	0 ... 65535	unsigned short int
long int	4	–2147483648 ... 2147483647	long, signed long int
unsigned long int	4	0 ... 4294967295	unsigned long
float	4	±(3.4E–38 ... 3.4E+38)	нет
double	8	±(1.7E–308 ... 1.7E+308)	нет
long double	10	±(3.4E–4932 ... 1.1E+4932)	нет

АРИФМЕТИЧЕСКИЕ ТИПЫ ДАННЫХ ЯЗЫКА СИ++

Замечания по типам данных:

- если не указан базовый тип, то по умолчанию это **int**
- если не указан модификатор знаков, то по умолчанию **signed**
- с базовым типом **float** модификаторы не употребляются
- модификатор **short** применяется только к базовому типу **int**
- в СИ и СИ++ величины типа **char** могут рассматриваться в программе и как **символы**, и как **целые числа**, в зависимости от контекста
- среди базовых типов нет логического типа данных
- в последние версии СИ++ добавлен отдельный логический тип **bool**

ПЕРЕМЕННЫЕ. ОПИСАНИЕ ПЕРЕМЕННЫХ В СИ И СИ++

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

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

Типы переменных

- int – целое число в интервале [-32768…32767] (2 байта)
- float – вещественное число, *floating point* (4 байта)
- char – символ, *character* (1 байт)

ПЕРЕМЕННЫЕ. ОПИСАНИЕ ПЕРЕМЕННЫХ В СИ И СИ++

Имена переменных

Могут включать

- латинские буквы (A-Z, a-z)
- знак подчеркивания _
- цифры 0-9

НЕ могут включать

- русские буквы
- пробелы
- скобки, знаки +, =, !, ? и др.

Какие имена правильные?

AXby R&B 4Wheel Вася “PesBarbos”
TU154 [QuQu] _ABBA A+B

Описание переменных

Описание переменных в Си и Си++ имеет вид

имя_типа список_переменных;

Одновременно с описанием можно задать начальные значения переменных. Такое действие называется **инициализацией** переменных.

Описание с инициализацией в Си и Си++ имеет вид

тип имя_переменной = начальное_значение;

Объявление переменных

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

```
void main()
{
i
f
int Tu104, Il86=23, Yak42;
float x=4.56, y, z;
char c, c2='A', m;
}
```

целая переменная a

вещественные
перемен

целая и дробная части отделяются точкой

целые переменные Tu104, Il86 и Yak42

вещественные переменные x, y и z
x = 4,56

символьные переменные c, c2 и m
c2 = 'A'

Если начальное значение не задано, в этой ячейке находится "мусор"!

КОНСТАНТЫ В СИ И СИ++

КОНСТАНТЫ В СИ И СИ++

Именованные константы (константные переменные)

Для их определения используется квалификатор доступа **const**, указывающий, что величина не может изменяться в течение всего времени работы программы.

Примеры:

```
const float pi = 3.14159;
const int i = 100, A = 1;
```

Определение констант на стадии препроцессорной обработки программы

Еще одна возможность ввести именованную константу – использование препроцессорной директивы **#define**. Тип констант явно не указывается и определяется по форме записи

Примеры:

```
#define pi  3.14159
#define i 100
```

ОПЕРАЦИИ И ВЫРАЖЕНИЯ В СИ И СИ++

Во всех языках программирования под **выражением** подразумевается конструкция, составленная из кон-стант, переменных, знаков операций, функций, скобок

Выражение определяет порядок вычисления некото-рого значения. Если оно числовое – то выражение **арифметическое.**

Примеры арифметических выражений:

традиционные для ЯПВУ	**специфичные для языка Си**
a + b	**x++**
2 * (X – Y)	**y++ * b**
12.5 / (k - 1)	**--n * 2**
	f += 4

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В СИ И СИ++

– вычитание (унарный минус)

+ сложение

*** умножение**

/ деление

% деление по модулю (остаток от деления)

++ инкремент (унарная операция +1)

– – декремент (унарная операция -1)

Замечание: операция % применима только к целым

Особенность операции деления в Си и СИ++

При делении целых чисел остаток отбрасывается и результат – целое число!

Иначе – результат = вещественное число!

```
void main()
{
int a = 7;
float x;
x = a / 4;
x = 4 / a;
x = float(a) / 4;
x = 1.*a / 4;
}
```

1

0

1.75

1.75

ОСОБЕННОСТИ ОПЕРАЦИЙ ИНКРЕМЕНТ И ДЕКРЕМЕНТ

- ✔ они могут применяться только к переменным и не могут – к константам и выражениям
- ✔ существует префиксная (++X) и постфиксная (X++) формы записи
- ✔ Эти операторы дадут один результат:

  X = X + 1 X++ ++X
- ✔ Различия – при использовании в выражениях:

Пример 1:

a =3; b=2;

c = a++*b++;

Результат:

a=4, b=3, c=6.

Пример 2:

a =3; b=2;

c = ++a*++b;

Результат:

a=4, b=3, c=12.

ПОРЯДОК АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

- По убыванию старшинства арифметические операции располагаются в следующем порядке:
  ++, - -
  - (унарный минус)
  ***, /, %**
  +, -

- Одинаковые по старшинству операции выполняются слева направо.
- Для изменения порядка выполнения операций в выражениях могут применяться круглые скобки

Порядок выполнения операций

- вычисление выражений в скобках
- умножение, деление, % слева направо
- сложение и вычитание слева направо

```
          2  3  5  4   1     7  8   6   9
z = (5*a*c+3*(c-d))/a*(b-c)/ b;
```

$$z = \frac{5ac + 3(c-d)}{ab}(b-c)$$

$$x = \frac{a^2 + 5c^2 - d(a+b)}{(c+d)(d-2a)}$$

```
        2  6  3  4  7  5   1     12   8 11 10  9
x =(a*a+5*c*c-d*(a+b))/((c+d)*(d-2*a));
```

ОПЕРАЦИИ ОТНОШЕНИЯ

- Набор операций отношений стандартный:
  <
  <
  <=
  >=
  ==
  !=
- Замечание: т.к. логического типа данных в стандарте языка Си нет, то результат операций отношения: если отношение истинно, то **1**, если отношение ложно, то **0**.
- Определите результат операций:
  101 > 105 ‘F’ == ‘f’ ‘a’ != b a<0

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

- Набор операций стандартный:
  ! **операция отрицания**
  && **конъюнкция (логическое И)**
  || **дизъюнкция (логическое ИЛИ)**

  Пример: выражение **0<x<1** примет вид **x>0 && x<1**

- Приоритет логических операций и операций отношений:
  !
  <, >, <=, >=
  ==, !=
  &&
  ||

ОПЕРАЦИИ ПРИСВАИВАНИЯ

- В языке Си и Си++ присваивание является операцией, а не оператором.
- Знак операции присваивания **=**
- Как любая другая операция, операция присваивания может входить в выражение несколько раз: **a=b=c=s+y-1**
- Присваивание имеет самый низкий приоритет среди операций
- Операция присваивания – **правоассоциативная**
- В языке Си и Си++ существуют **дополнительные операции присваивания**

Дополнительные операции присваивания (Сокращенная запись операций в Си)

полная запись	сокращенная запись
`a = a + 1;` (инкремент)	`a++;`
`a = a + b;`	`a += b;`
`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;`

ОПЕРАЦИЯ ЯВНОГО ПРЕОБРАЗОВАНИЯ ТИПА

Формат операции:
(имя_типа) операнд
Пример: (float) 1, (int) x%2

ОПЕРАЦИЯ sizeof

Форматы операции:
sizeof (тип) и sizeof (выражение)

Результатом является число, равное количеству байтов, которое занимает в памяти компьютера величина указанного типа или величина, полученная в результате вычисления выражения.

Пример: sizeof (float) равно 4, sizeof (5%2) равно 2

ОПЕРАЦИЯ «ЗАПЯТАЯ»

Эта операция используется для связывания нескольких выражений в одно.

ОПЕРАЦИЯ «УСЛОВИЕ ? :»

Эта единственная операция, имеющая 3 операнда. она реализует алгоритмическую структуру «ветвление». Формат операции:

выражение 1 ? выражение 2 : выражение 3

Примеры: |X| X<0 ? –X : X;

max(a, b) max = (a<=b) ? b : a;

ОПЕРАЦИИ ( ) И [ ]

Приоритеты (ранги) операций в Си++

Ранг	Операции	Ассоциа-тивность
1	( ) [ ] –> .	→
2	! ~ + – ++ – – & * (тип) sizeof (унарные)	←
3	* / % (мультипликативные бинарные)	→
4	+ – (аддитивные бинарные)	→
5	<< >> (поразрядного сдвига	→
6	< <= >= > (отношения)	→
7	= = != (отношения)	→
8	& (поразрядная конъюнкция «И»)	→
9	^ (поразрядное исключающее «ИЛИ»)	→
10	| (поразрядная дизъюнкция «ИЛИ»)	→
11	&& (конъюнкция «И»)	→
12	| | (дизъюнкция «ИЛИ»)	→
13	?: (условная)	←
14	= *= / = %= + = – = &= ^= | = <<= >>=	←

ПРИВЕДЕНИЕ ТИПОВ ПРИ ВЫЧИСЛЕНИИ ВЫРАЖЕНИЙ

Все ЯПВУ имеют ряд общих правил записи выражений

- ✔ все символы, составляющие выражение, записываются в строку
- ✔ в выражении проставляются все знаки операций
- ✔ при записи выражения учитываются приоритеты операций
- ✔ для влияния на последовательность операций используются круглые скобки

В процессе вычисления выражений с разнотипными операндами производится автоматическое преобразование типов величин:

- преобразование не выполняется, если оба операнда одного типа
- при разных типах операндов происходит приведение величины с младшим типом к старшему типу (кроме

ПРИВЕДЕНИЕ ТИПОВ ПРИ ВЫЧИСЛЕНИИ ВЫРАЖЕНИЙ

Старшинство типов друг по отношению к другу определя-ется по следующему принципу: ***старший тип включает в себя все значения младшего типа как подмножество.***

Вещественные типы являются старшими по отношению к целым.

Внутри каждой из этих 2 групп типов иерархия устанавливается в соответствии с указанным принципом.

Целые типы по возрастанию старшинства:

char → shot → int → long

Вещественные типы по возрастанию старшинства:

float → double→ long double

Задание на тему «Элементы языка Си++. Типы данных. Операции и выражения»

Задание на тему «Элементы языка Си++. Типы данных. Операции и выражения».

1. Какие последовательности символов не являются лексемами языка Си++ **(зачеркнуть!)**

а) B12 б) + в) _ngf г) ″Си″ д) c-d е) else ж) Flag з) ris_5 и) 23E–7 к) 12_vf л) 34 м) My_pen

2. Какие последовательности символов не являются идентификаторами языка Си++ **(зачеркнуть!)**

а) B12 б) + в) My pen г) ″Си″ д) c-d е) else ж) Flag з) ris_5 и) 23E–7 к) 12_vf л) 34 м) My_pen

3. Определите тип константы **(подпишите рядом с константой)**

а) 312 б) –32.4 в) 102408 г) 0315 д) 0x24 е) 6.2L

ж) ′5′ з) ′F′ и) ′ \n ′ к) ″ My pen ″ л) 23E–7

4. В программе объявлена переменная **int n=10.** Определите значение выражения **(подпишите сверху)**

а) n = 312 + n; б) – – n в) + + n + 5 г) + + n д) n / 3 е) n / 4. ж) 5 + n + + з) n < 0

5. Координаты точки на плоскости **(X, Y).** Запишите в форме логических выражений **(подпишите рядом)**

а) точка лежит в первой четверти координатной плоскости

б) точка лежит на единичной окружности

в) точка лежит на одной из координатных осей

6. Объявите переменные, необходимые для решения задачи **(подпишите рядом, вставьте комментарий)**

а) вычисление площади прямоугольника

Линейные программы на Си/Си++

Тема 20

Пример линейной программы на Си

```
/*Пример программы на Си/Си++  */
#include <stdio.h>
#include <math.h>
#include <conio.h>
void main ()
{
float a,b,c,p,s;
printf("\n a="); scanf("%f", &a);
printf("\n b="); scanf("%f", &b);
printf("\n c="); scanf("%f", &c);
p=(a+b+c)/2;
s=sqrt(p*(p-a)*(p-b)*(p-c));
printf("\n Площадь треугольника=%4.1f", s);
getch();
}
```

директивы препроцессора

объявление переменных

исполняемые операторы

Исполняемые операторы

Оператор – это команда языка программирования высокого уровня.

Понятие «**оператор**» в языке Си трактуется следующим образом – **любое выражение, после которого стоит точка с запятой, воспринимается компилятором как отдельный оператор.** Оператор определяет законченное действие на очередном шаге выполнения программы.

Поэтому такая конструкция языка Си тоже является оператором:

i ++; ***это оператор-выражение.***

Если вычисление выражения заканчивается присваиванием, то его называют **оператором присваивания.**

Линейная программа может содержать только

Оператор присваивания

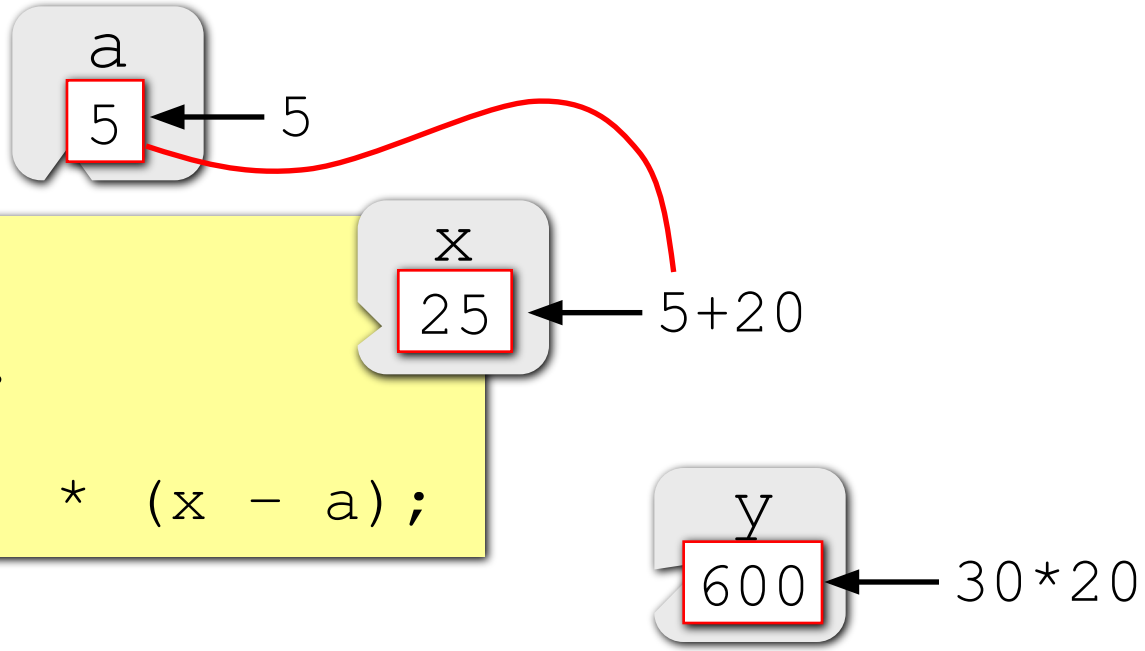
Оператор присваивания служит для изменения значения переменной.

Пример

```
a = 5;

x = a + 20;

y = (a + x) * (x - a);
```



Оператор присваивания

Общая структура:

куда

что

```
имя переменной = выражение;
```

Арифметическое выражение может включать

- константы (постоянные)
- имена переменных
- знаки арифметических операций:

```
+  -     *     /     %
```

умножение

деление

остаток от деления

- вызовы функций
- круглые скобки ( )

? Для чего служат круглые скобки?

Какие операторы неправильные?

```
void main()
{
   int a, b;
   float  x, y;
  a = 5;
   10 = x;
   y = 7,8;
   b = 2.5;
   x = 2*(a + y);
   a = b + x;
}
```

имя переменной должно быть слева от знака =

целая и дробная часть отделяются точкой

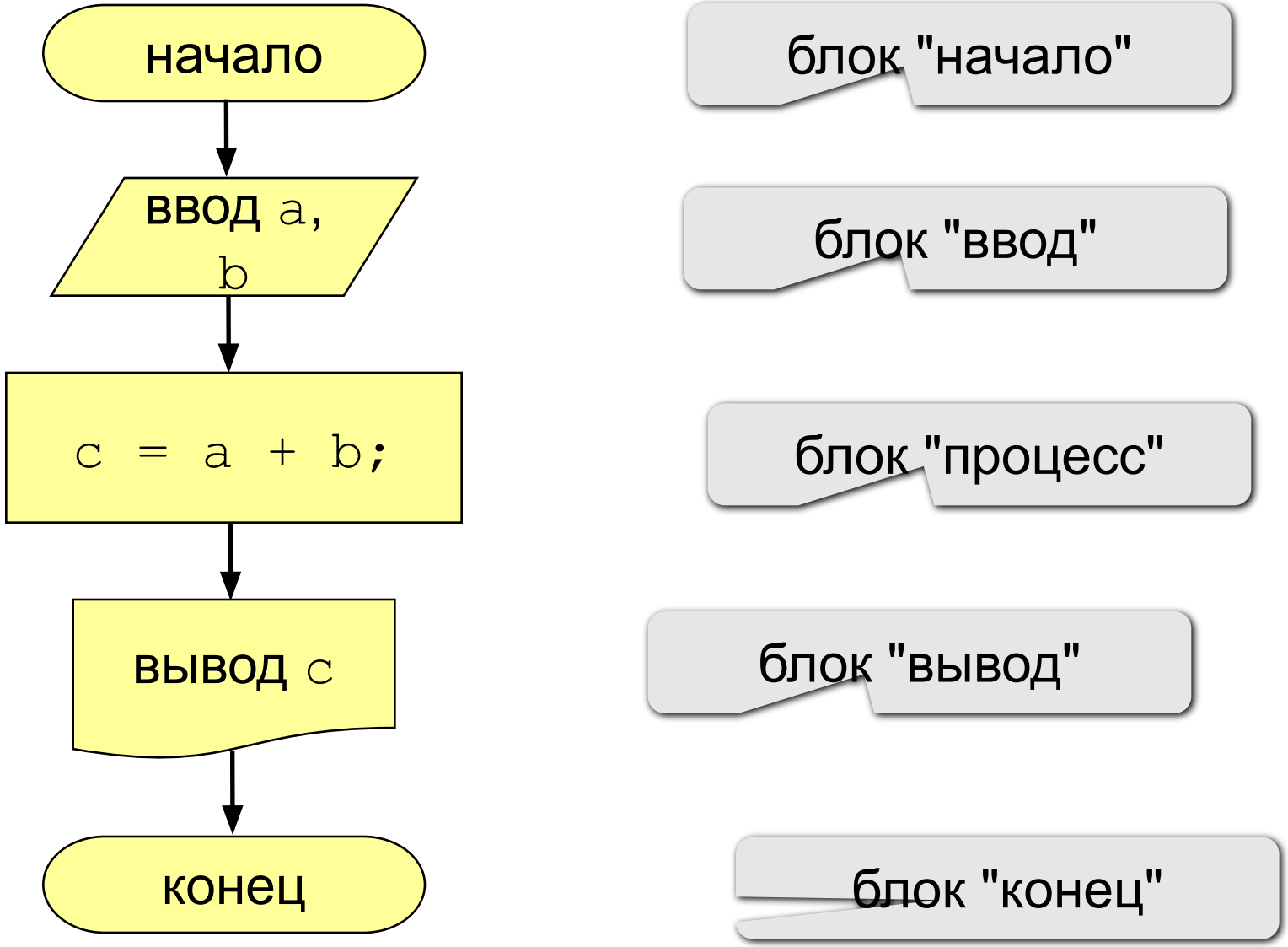
при записи вещественного значение в целую переменную дробная часть будет отброшена

Библиотека математич. функций

Обращение	Тип арг-та	Тип рез-та	функция
abs(x)	int	int	абс.знач. целого ч.
cos(x)	double	double	косинус (x в рад.)
exp(x)	double	double	e^x
fabs(x)	double	double	абс.знач. веществ. ч.
log(x)	double	double	ln(x)
log10(x)	double	double	$\log_{10}(x)$
pow(x, y)	double	double	x^y
sin(x)	double	double	синус (x в рад.)
sqrt(x)	double	double	корень квадратный

Сложение двух чисел: Ввести два целых числа и вывести на экран их сумму.

Блок-схема линейного алгоритма



Сложение двух чисел

Задача. Ввести два целых числа и вывести на экран их сумму.

Простейшее решение:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b, c;
  printf("Введите два целых числа\n");
  scanf ("%d%d", &a, &b);
  c = a + b;
  printf("%d", c);
  getch();
}
```

подсказка для ввода

ввод двух чисел с клавиатуры

вывод результата

Форматированный ввод/ вывод

Операторы **printf()** и **scanf()** обеспечивают ввод данных с клавиатуры и вывод на экран.

Они осуществляют обращение к соответствующим функциям стандартной библиотеки ввода-вывода Си, заголовочный файл которой имеет имя stdio.h.

Структура оператора вывода на экран:

printf(форматная_строка, список_аргументов);

Форматная строка ограничена кавычками (т.е. является текстовой константой) и может включать в себя **произво-льный текст, управляющие символы и спецификаторы формата.**

Форматированный вывод на экран

printf – форматный вывод

формат вывода

список выводимых значений

```
printf("\n результат=%f",S);
```

управляющий символ

выводимый текст

Признак управляющего символа ****

Список управляющих символов:

\n – перевод строки

\t – горизонтальная табуляция

\a – звуковой сигнал

\r, \v. \f, \b

Форматированный вывод на экран

printf – форматный вывод

формат вывода

список выводимых значений

```
printf("\n результат=%f",S);
```

управляющий символ

выводимый текст

Спецификатор формата это пара символов, начинающа-яся с %, определяющая форму вывода на экран.

Список некоторых спецификаторов формата:

% c – символ, % s – строка,

% d – целое десятичное число (типа int)

% u - целое десятичное число без знака (типа unsigned)

% f – вещественные числа в формате с фиксир.

Форматированный вывод на

```
printf("\n результат=%4.1f",S);
```

К спецификатору формата могут быть добавлены числовые параметры: ***ширина поля и точность.*** ***Ширина –*** это число позиций, отводимых на экране под величину, а ***точность*** – число позиций под дробную часть.

Эти параметры записываются между значком % и символом формата и разделяются точкой. Если выводимое значение не помещается в пределы указанной ширины, то этот параметр будет игнорироваться и величина будет выводиться полностью.

К спецификаторам формата могут быть добавлены модификаторы:

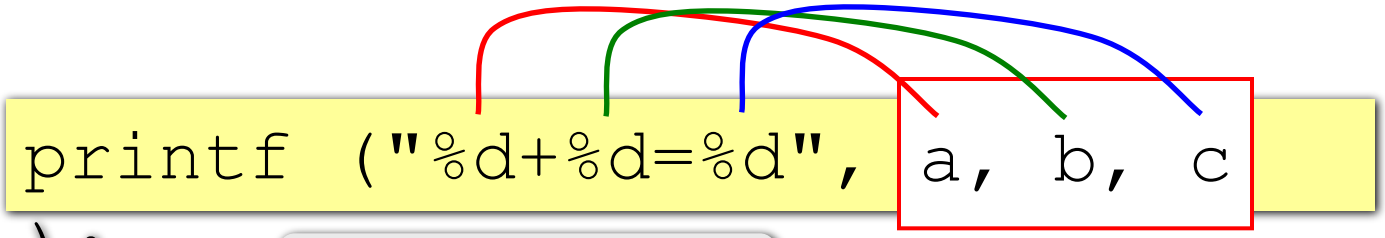
Вывод чисел на экран

здесь вывести целое число

это число взять из ячейки c

```
printf ("%d", c);
```

```
printf ("Результат: %d", c);
```



```
printf ("%d+%d=%d", a, b, c
);
```

формат вывода

список значений

```
printf ("%d+%d=%d", a, b, a+b
);
```

арифметическое выражение

Вывод целых чисел

```
int x = 1234;
printf ("%d", x);
```

или "%i"

```
1234
```

минимальное число позиций

или "%9i"

```
printf ("%9d", x);
```

```
     1234
```

всего 9 позиций

Вывод вещественных чисел

```
float x = 123.4567;
printf ("%f", x);
```

```
123.456700
```

минимальное число позиций, 6 цифр в дробной части

```
printf ("%9.3f", x);
```

```
  123.456
```

всего 9 позиций, 3 цифры в дробной части

```
printf ("%e", x);
```

```
1.234560e+02
```

стандартный вид: $1{,}23456 \cdot 10^2$

```
printf ("%10.2e", x);
```

```
  1.23e+02
```

всего 10 позиций, 2 цифры в дробной части мантиссы

Ввод чисел с клавиатуры

scanf – форматный ввод

формат ввода

адреса ячеек, куда записать введенные числа

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

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

- `%d` – целое число
- `%f` – вещественное число
- `%c` – 1 символ
- `%s` – символьная строка

&a – адрес переменной a

```
7652
12
```

a – значение переменной a

ждать ввода с клавиатуры двух целых чисел (через пробел или *Enter*), первое из них записать в переменную a, второе – в b

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

&a

%d%d

&a, &b

убрать пробел

%d%d

Полное решение

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b, c;
  clrscr();
  printf("Введите два целых числа\n");
  scanf("%d%d", &a, &b);
  c = a + b;
  printf("%d+%d=%d", a, b, c);
  getch();
}
```

это выводит компьютер

Протокол:

```
Введите два целых числа
25 30
25+30=55
```

это вводит пользователь

Потоковый ввод/ вывод

В Си++ имеются свои специфические средства ввода данных с клавиатуры и вывода на экран. Это ***библиотека классов***, подключаемая к программе с помощью файла ***iostream.h.*** В этой библиотеке определены в качестве объектов стандартные символьные потоки с именами:

cin – стандартный поток ввода с клавиатуры

cout – стандартный поток вывода на экран.

Ввод данных интерпретируется как ***извлечение из стандартного потока cin b*** и присваивание вводимых значений соответствующим переменным. В Си++ опреде-лена операция извлечения из стандартного потока, знак которой **>>**.

Структура оператора ввода: **cin>>x;**

Потоковый ввод/ вывод

Вывод данных интерпретируется как ***помещение в стандартный поток cout*** выводимых значений.

Выводиться могут тексты, заключенные в двойные кавычки, и значения выражений. Знак операции помещения в поток **<<**.

Примеры оператора вывода на экран:

```
cout<< x;
cout<<a+b;
cout<<“\nРезультат=”<<Y;
cout<<“x=”<<x<<“   y=”<<y<<“\tz=”<<z<<endl;
```

Видим, что можно использовать управляющие символы. Элемент вывода **endl** – манипулятор, аналогичный **\n**.

Потоковый ввод/ вывод

В процессе потокового ввода-вывода происходит преобразование из формы внешнего символьного представления во внутренний формат и обратною

Тип данных и необходимый формат определяются автоматически. Стандартные форматы задаются специальными флагами форматирования, которые устанавливаются с помощью функции setf().

Кроме того, на формат отдельных выводимых данных можно влиять путем применения специальных манипуляторов. Но эти вопросы рассматривать не будем.

Пример линейной программы на Си++

```
//Пример программы на Си++
#include <iostream.h>
#include <math.h>
#include <conio.h>
void main ()
{
float a,b,c,p,s;
cout<<"\n a=";  cin>>a;
cout<<"\n b=";  cin>>b;
cout>>"\n c=";  cin>>c;
p=(a+b+c)/2;
s=sqrt(p*(p-a)*(p-b)*(p-c));
cout<<"\n Площадь треугольника=“<< s;
getch();
}
```

директивы препроцессора

объявление переменных

исполняемые операторы

Задачи на линейные алгоритмы

Написать программы на Си++ для реализации линейных алгоритмов

- **Задача 1:** Дан радиус окружности, найти длину окружности и площадь круга
- **Задача 2**: Даны три точки на плоскости (x_1, y_1), (x_2, y_2), (x_3, y_3). Найти периметр и площадь получившегося треугольника.
- **Задача 3**: X тетрадей и Y ручек стоят Z рублей, причем известно, что тетрадь на 2 рубля дороже ручки. Сколько стоит 1 ручка и сколько стоит 1 тетрадь?

Домашняя работа на линейные алгоритмы

Написать программы на Си++ для реализации линейных алгоритмов

- **Задача 1:** «Геометрическая задача» - по вариантам (по карточкам).

- **Задача 2**: «Физическая задача» - по вариантам (по карточкам).