Базовые понятия языка С

доцент КИ Наталия Викторовна Голкова

Литература

- Т.В. Нестеренко, Т.Г.Чурина. Учебное пособие “Основы программирования”. Новосибирск: ВКИ НГУ, 2015.
- Т.В. Нестеренко. Методическое пособие к курсу «Методы программирования» (часть 1) Лабораторные работы. Новосибриск, 2008
- Брайан Керниган, Деннис Ритчи. Язык программирования Си.
- Хэзфилд Р., Кирби Л. Искусство программирования на С: Фундаментальные алгоритмы, структуры данных и примеры приложений (пер. с англ.) {Энциклопедия программиста} К: ДиаСофт`01- 736 с.
- Подбельский В.В., Фомин С.С. Программирование на языке Си: учеб.пособие. –М.: Финансы и статистика, 2007.-600с.
- Павловская Т.А. С/С++. Программирование на языке высокого уровня – СПб.: Питер, 2005.- 461 с.
- Вирт Н. Алгоритмы и структуры данных. М.: Мир, 1989.–360с.
- А.Ахо, Д.Хопкрофт, Д. Ульман. Структуры данных и алгоритмы. М.: Издательский дом “Вильямс”, 2000 – 384 с..

Дифференцируемый зачет автоматом

Опоздание или уход со второй части пары = **Н**

5 мин перерыв между уроками - **не допускается опоздание по причине перекура или похода за кофе**
Вход с напитками и булками в аудиторию запрещен. Можно бутылочку воды.

Основные понятия программирования

Программирование — это процесс создания компьютерной программы, **теоретический** и **практический**.

1. Теоретический.

Алгоритмизация — **разработка** общего **метода решения** задачи.

2. Практический.

Кодирование — **написание программ** на **языке программирования**.

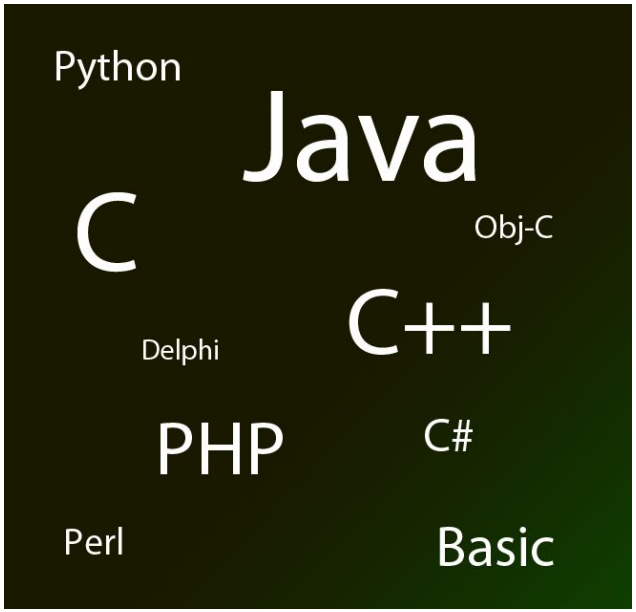
Программа - это последовательность команд (**алгоритм**), понятных **компьютеру.**

Язык программирования – это **набор правил** записи алгоритмов и данных.

Величины, с которыми работает компьютер называют **данными.**

Язык программирования

Язык программирования – это **способ «сказать» компьютеру** что он должен выполнять.



Введение в язык программирования Си

ANSI C — стандарт языка Си, опубликованный Американским национальным институтом стандартов (ANSI).

Следование этому стандарту помогает создавать легко портируемые программы.

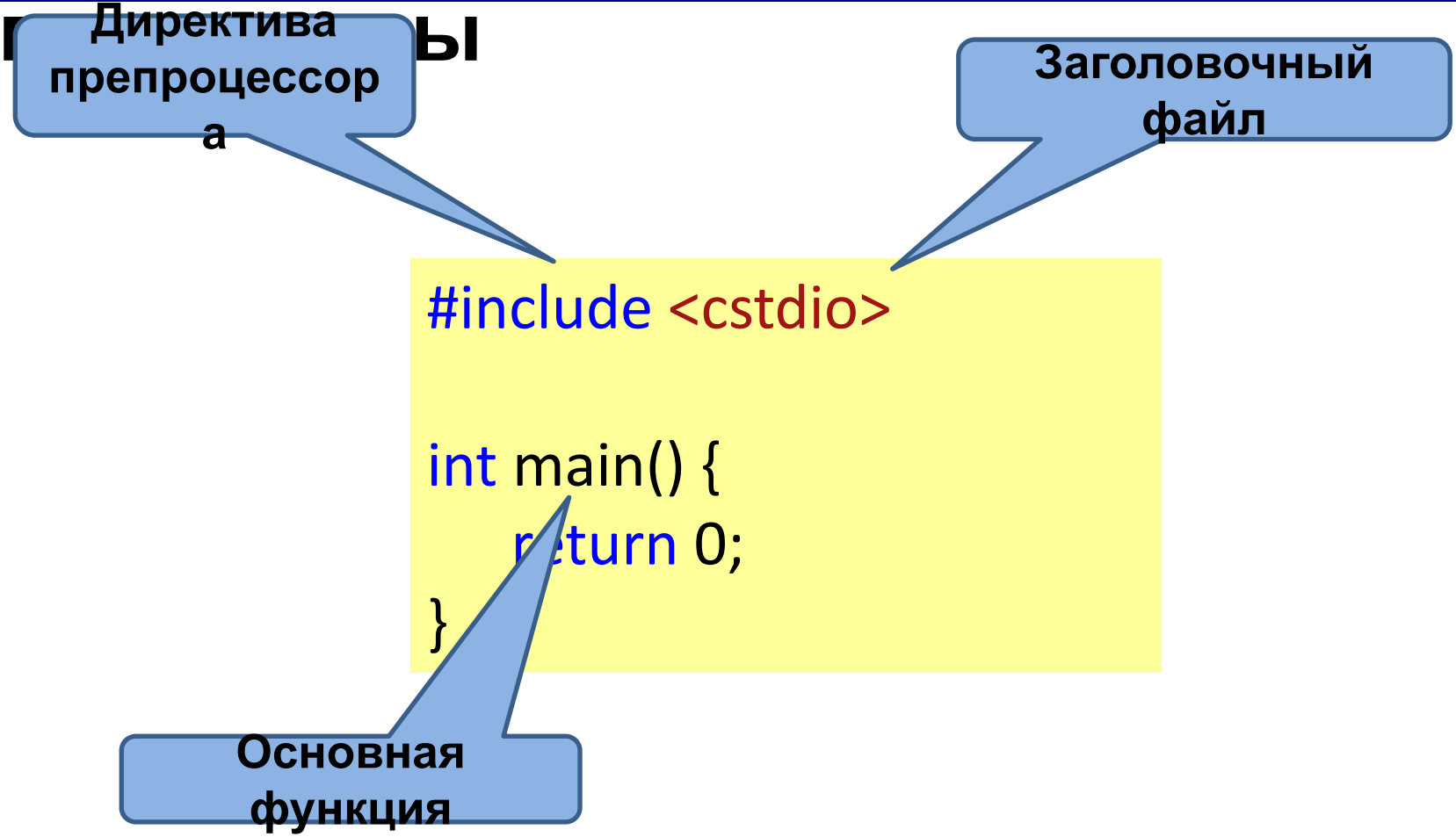
Введение в язык программирования Си

Любая программа состоит из **функций** и **переменных**.

Функции содержат **команды** которые **выполняет компьютер**.

Переменные **хранят значения**, используемые в процессе вычислений.

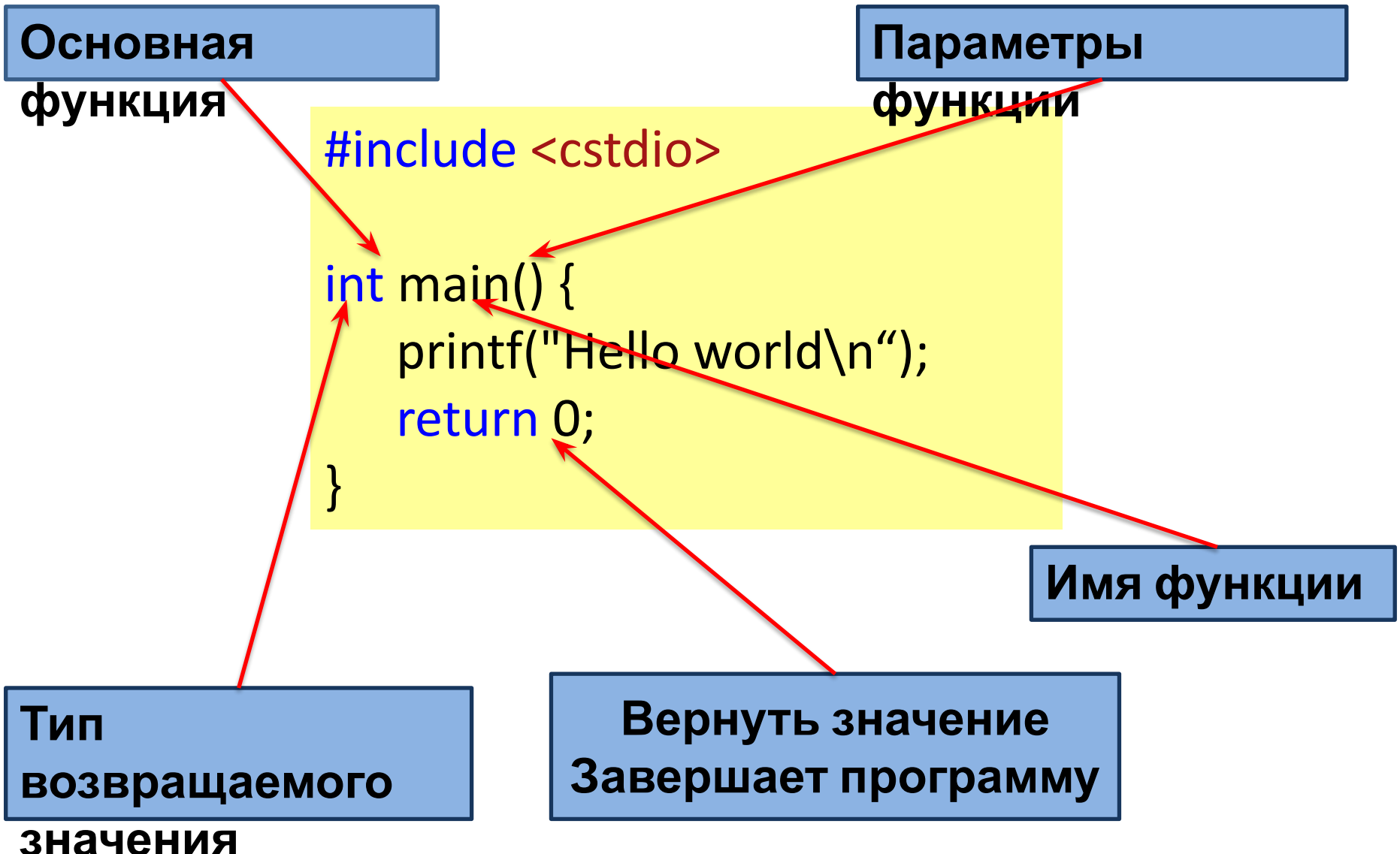
Структура программы



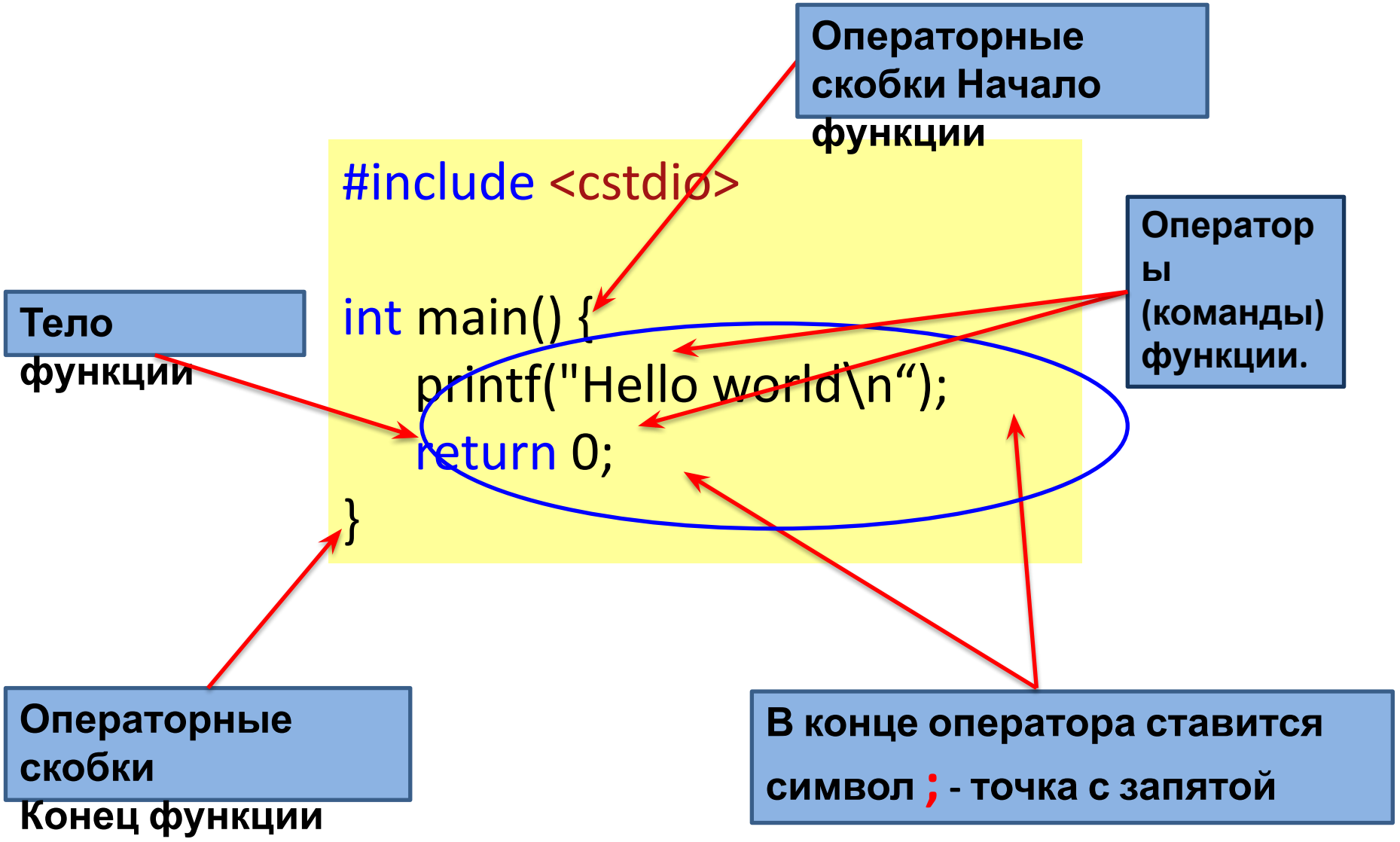
- диез

Директива #include просто целиком подставляет файл, который передан параметром директиве.

Структура программы



Структура программы



Комментарии

Комментарии помогают **понять код** программы.
Комментарии не включаются в содержимое исполняемого файла и не влияют на исполнение программы.
Компилятор **игнорирует** все, что помечено в программе как **комментарий**.

```
// строка комментария
/* блок
   комментария */
```

```
#include <cstdio>  // директива препроцессора
int main() {//начало тела функции main
      printf("Hello world\n ");  //выводит на экран “Здравствуй мир”
      return 0;    //Вернуть значение
}   // конец тела функции
```

Переменные

Пример переменной в математике.
Вычислить значение **выражения** 236 + x, если $x =$ 364, 870, 17
Решение:

236 + x = 236 + 364 = 600

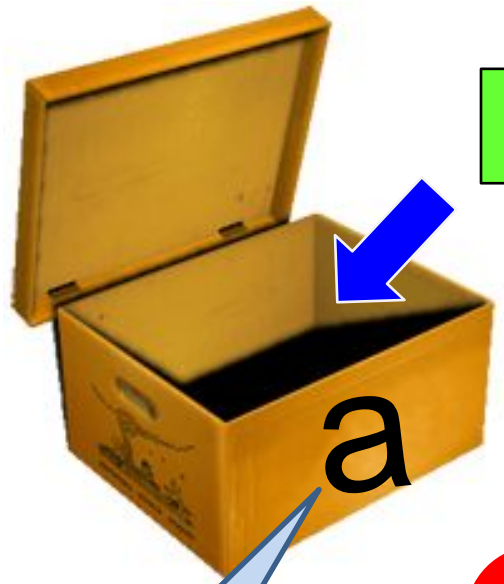
236 + x = 236 + 870 = 1106

236 + x = 236 + 17 = 253

Буквы, которые входят в математические выражения и которые могут **принимать разные численные значения**, называются **переменными**.

Переменные (variable)

Переменная в программировании – это величина, имеющая **имя**, **тип** и **значение**. **Значение** переменной можно **изменять** во время работы программы.



Имя

Значение

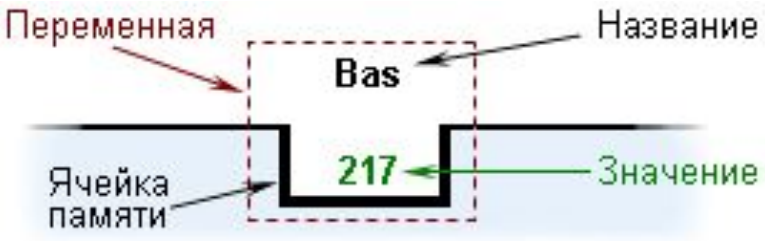
Другой тип данных

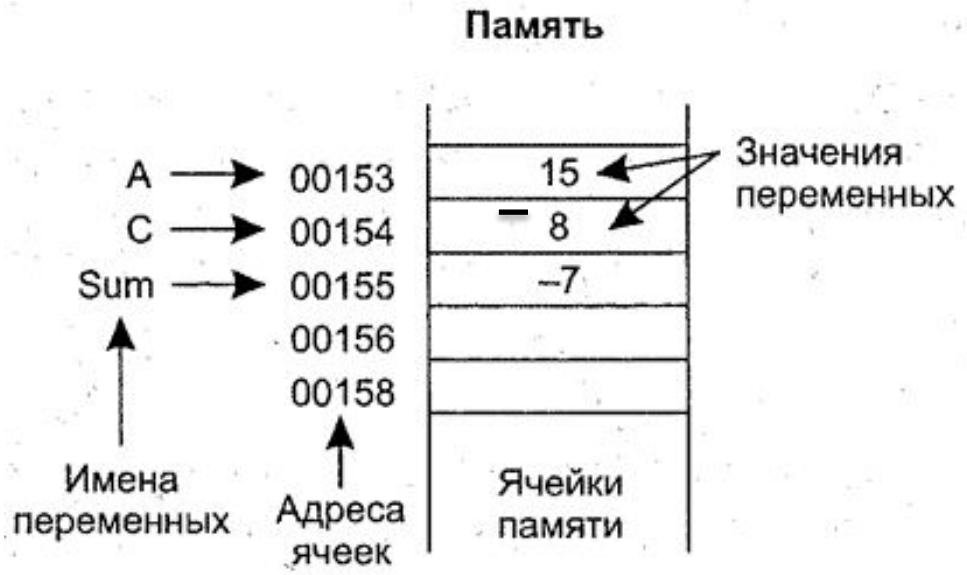
Поместится?

В переменной хранятся данные определенного типа!

Переменные (variable)

Переменные хранятся в определенных «**ячейках**» памяти компьютера.





Имена переменных

В именах МОЖНО использовать БЕЗ ПРОБЕЛА

- латинские буквы (A-Z)

  заглавные и строчные буквы различаются, z и Z разные имена переменных

- цифры

  имя не может начинаться с цифры

- знак подчеркивания _

В именах НЕЛЬЗЯ использовать

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

Какие имена правильные??

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

Имена переменных

Термин "идентификатор переменной" тоже что и "имя переменной"

"идентификатор переменной" = "имя переменной"

Типы данных

Тип данных определяет

- *множество значений*,
- *набор операций*, которые можно применять к таким значениям,
- возможно, *способ реализации хранения значений и выполнения операций*.

Типы данных

Типы данных: простые и структурированные.

Простые - это целые и вещественные числа, символы (относятся к целым), перечислимые и указатели (адреса объектов в памяти).

Целые, символы и перечислимые – порядковые типы.

Структурированные данные – это массивы и структуры.

Типы данных

Базовые типы:

1) `int` - целый;
2) `char` - символьный;

} целые

3) `float` - вещественный;
4) `double` – вещ. двойной точности;

} вещественные

5) `void` – не имеющий значения.

Характеристика базовых типов

1. Переменная типа *char* обычно имеет размер 1 байт, ее значениями являются различные символы из кодовой таблицы, например: 'ф', ':', 'j' (при записи в программе они заключаются в одинарные кавычки).

2. Размер переменной типа *int* в стандарте языка Си не определен. В большинстве систем программирования размер переменной типа *int* соответствует размеру целого машинного слова. Например, в компиляторах для 16-разрядных процессоров переменная типа *int* имеет размер 2 байта. В этом случае знаковые значения этой переменной могут лежать в диапазоне от -32768 до 32767. В современных компьютерах – 4 и 8 байт.

Характеристика базовых типов

3. Ключевое слово **`float`** позволяет определить переменные вещественного типа. Их значения имеют дробную часть, отделяемую точкой, например: **`-5.6, 31.28`** и т.п.

   Вещественные числа могут быть записаны также в форме с плавающей точкой, например: **`1.09e+4`**.

   Число перед символом "e" называется мантиссой, а после "e" – порядком или экспонентой. Переменная типа **`float`** занимает в памяти 4 байта. Она может принимать значения в диапазоне

   от **`3.4e-38 до 3.4e+38`**.

Характеристика базовых типов

4. Ключевое слово **double** позволяет определить вещественную переменную двойной точности. Она занимает в памяти в два раза больше места, чем переменная типа **float**. Переменная типа **double** может принимать значения в диапазоне от **1.7e-308** до **1.7e+308.**
5. Ключевое слово **enum** позволяет определить переменную перечислимого типа.

6. Ключевое слово **void** используется для нейтрализации значения объекта, например, для объявления функции, не возвращающей никаких значений.

Большие числа

Имеются еще целые типы, имена которых начинаются с символов «__int», за которыми следует число бит. При записи констант этих типов можно использовать суффиксы i и ui, как показано в приведенной ниже таблице

Тип	Суффикс	Пример	Память (биты)
_int8	i8	_int8 c = 127i8;	8
_int16	i16	_int16 s = 32767i16;	16
_int32	i32	_int32 i = 12345678i32;	32
_int64	i64	_int64 big = 1234565432i64;	64
unsigned _int64	ui64	unsigned _int64 hugeInt = 1234567887654321ui64;	64

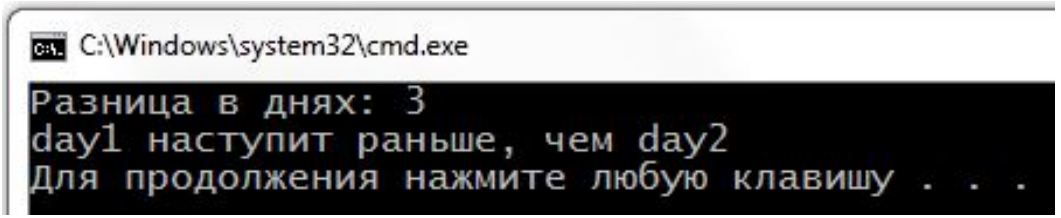
Перечисление

Перечислимый тип определяется как набор идентификаторов, с точки зрения языка играющих ту же роль, что и обычные именованные константы, но связанные с этим типом. Переменная, которая может принимать значение из некоторого списка значений, называется переменной перечислимого типа или перечислением.

```
enum week {
  Monday,       /* 0  (= 6)*/
  Tuesday,      /* 1 */
  Wednesday,    /* 2 */
  Thursday,
  Friday,
  Saturday,
  Sunday
} rab_ned ;
rab_ned =  Sunday;
```

Пример перечисления

```
// применение перечислений
#include <cstdio>
#include <locale>
// объявление перечисляемого типа
enum days_of_week { Sun, Mon, Tue, Wed, Thu, Fri, Sat };
int main() {
  setlocale(LC_ALL,"Russian");
  days_of_week day1, day2; // определения переменных,
  // хранящих дни недели
  day1 = Mon; // инициализация переменных
  day2 = Thu;
  int diff = day2 - day1; // арифметическая операция
  printf("Разница в днях: %d\n", diff);
  if(day1 < day2) {// сравнение
     printf("day1 наступит раньше, чем day2\n");
  }
  return 0;
}
```



Типы данных

Модификаторы:

1) `const` - запрещает изменять значение переменной в ходе программы
2) `unsigned` - положительные значения от 0
3) `signed` - все значения обычной переменной, можно не указывать
4) `long` - удлиняет диапазон значений переменной
5) `short` - укорачивает диапазон значений переменной

Типы данных

Модификаторы записываются перед спецификаторами типа, например:

unsigned char. Если после модификатора опущен спецификатор, то компилятор предполагает, что этим спецификатором является **int**.

Таким образом, следующие строки:

```
long a;
long int a;
```

являются идентичными и определяют объект ***a*** как длинный целый.

Диапазон типов данных

Тип	байт	Диапазон принимаемых значений
Целочисленные типы данных		
short int	2	-32 768 .. 32 767
unsigned short int	2	0 .. 65 535
int	4	-2 147 483 648 .. 2 147 483 647
unsigned int	4	0 .. 4 294 967 295
char	1	-128 .. 127
long int	4	-2 147 483 648 .. 2 147 483 647
unsigned char	1	0 .. 255
unsigned long int	4	0 .. 4 294 967 295

Единица измерения информации – **байт**.

1 байт = 8 бит

Диапазон типов данных

Тип	байт	Диапазон принимаемых значений
типы данных с плавающей точкой		
float	4	$3.4*10^{-38}$ .. $3.4*10^{38}$
long float	8	$1.7*10^{-308}$ .. $1.7*10^{308}$
double	8	$1.7*10^{-308}$ .. $1.7*10^{308}$

Константы

- *вещественные*, например 123.456, 5.61e-4. Они могут снабжаться суффиксом F или f, например 123.456F, 5.61e-4f;
- *целые*, например 125;
- *короткие целые*, в конце записи которых добавляется суффикс H или h, например 275h, 344H;
- *длинные целые*, в конце записи которых добавляется суффикс L или l, например 361327L;
- *беззнаковые*, в конце записи которых добавляется суффикс U или u, например 62125U;
- *восьмеричные*, в которых перед первой значащей цифрой записывается 0, например 071;
- *шестнадцатеричные*, в которых перед первой значащей цифрой записывается два символа 0x, например 0x5F;

Константы

- *символьные* - единственный символ, заключенный в одинарные кавычки, например **`'O', '2', '.'`** и т.п. Символы, не имеющие графического представления, можно записывать, используя специальные комбинации, например \n (код 10), \0 (код 0). Допускается и шестнадцатеричное задание кодов символов, которое представляется в виде: **`'\x2B', '\x36'`** и т.п.;
- *строковые* - последовательность из нуля символов и более, заключенная в двойные кавычки, например: "Это строковая константа". Кавычки не входят в строку, а лишь ограничивают ее. Строка представляет собой массив из перечисленных элементов, в конце которого помещается байт с символом '\0'. Таким образом, число байтов, необходимых для хранения строки, на единицу превышает число символов между двойными кавычками;
- *константное выражение*, состоящее из одних констант, которое вычисляется во время трансляции (например: **`a = 60 + 301`**);
- тип *long double*, в конце записи которых добавляется буква L или l, например: **`1234567.89L`**.

Примеры

```
const long int k = 25;
const m = -50;  /* подразумевается
     const int m = -50 */
const n = 100000;  /*подразумевается
     const long int n = 100000 */
```

Представление данных в памяти компьютера

Единица измерения информации – **байт**.

1 байт = 8 бит

1 бит – это **0** или **1**

Информация представляется в двоичном коде.

Например, число 17 – э

Спецификаторы

1) **`int`** - %i, %d;
2) **`char`** - %c;
3) **`float`** - %f;
4) **`double`** – %f (%F), (%lf (%lF) для scanf()), %g %G, %e
5) **`long double`** – %Lf, %LF, %Lg, %LG, %Le, %LE
5) **`unsigned int`** – %u
6) **`unsigned long long int`** – %llu
7) **`Строка – массив символов`** – %s

Переменные

Объявление переменных:

Выделение места в памяти

тип – целые

```
int a, b, c;
```

список имен переменных

Переменные всегда объявляются до использования!
Нельзя называть несколько переменных

Как записать значение в переменную?

Оператор присваивания

```
a = 5;
```

При записи нового значения старое стирается!

Оператор – это команда языка программирования (инструкция).

Оператор присваивания – это команда для записи нового значения в переменную. Значение записывается в ячейку памяти.

Присваивать переменной можно только то значение, которое хранит её тип!

Инициализация переменной

Объявление и определение переменной

```
int a = 5;
int m = n = 1;
int sum1 = sum2 = 0;
char b = 'k';
float x = 2.4;
```

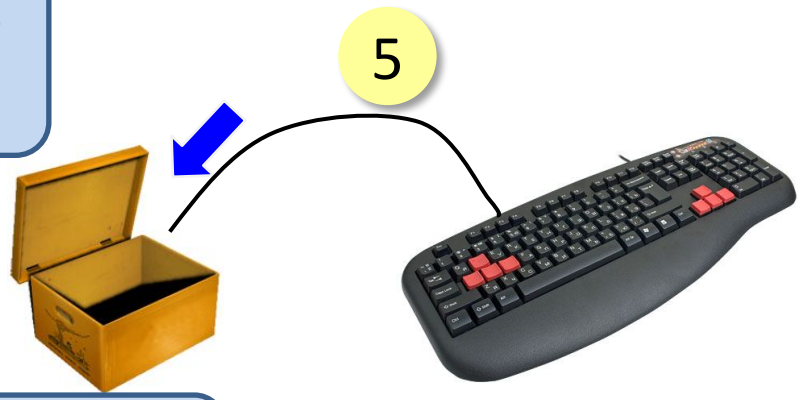
Как ввести значение с к

Оператор ввода

Спецификатор типа

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

Адрес переменной



1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную **a**.

Оператор вывода

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

/* вывод значения целой переменной a */

```
printf("%d\n",a);
```

/* вывод значения целой переменной **a** и переход на следующую строку*/

Оператор вывода

```
printf("Hello world!");  // вывод текста
```

```
printf("The result=%d",sum);
```

```
//вывод текста и значения переменной sum
```

Формат вывода значения (`printf`)

%d — печать десятичного целого.
%6d — печать десятичного целого в поле из шести позиций.
%f — печать числа с плавающей точкой.
%6f — печать числа с плавающей точкой в поле из шести позиций.
%.2f — печать числа с плавающей точкой с двумя цифрами после десятичной точки.
%6.2f — печать числа с плавающей точкой и двумя цифрами после десятичной точки в поле из шести
позиций.

Линейный алгоритм

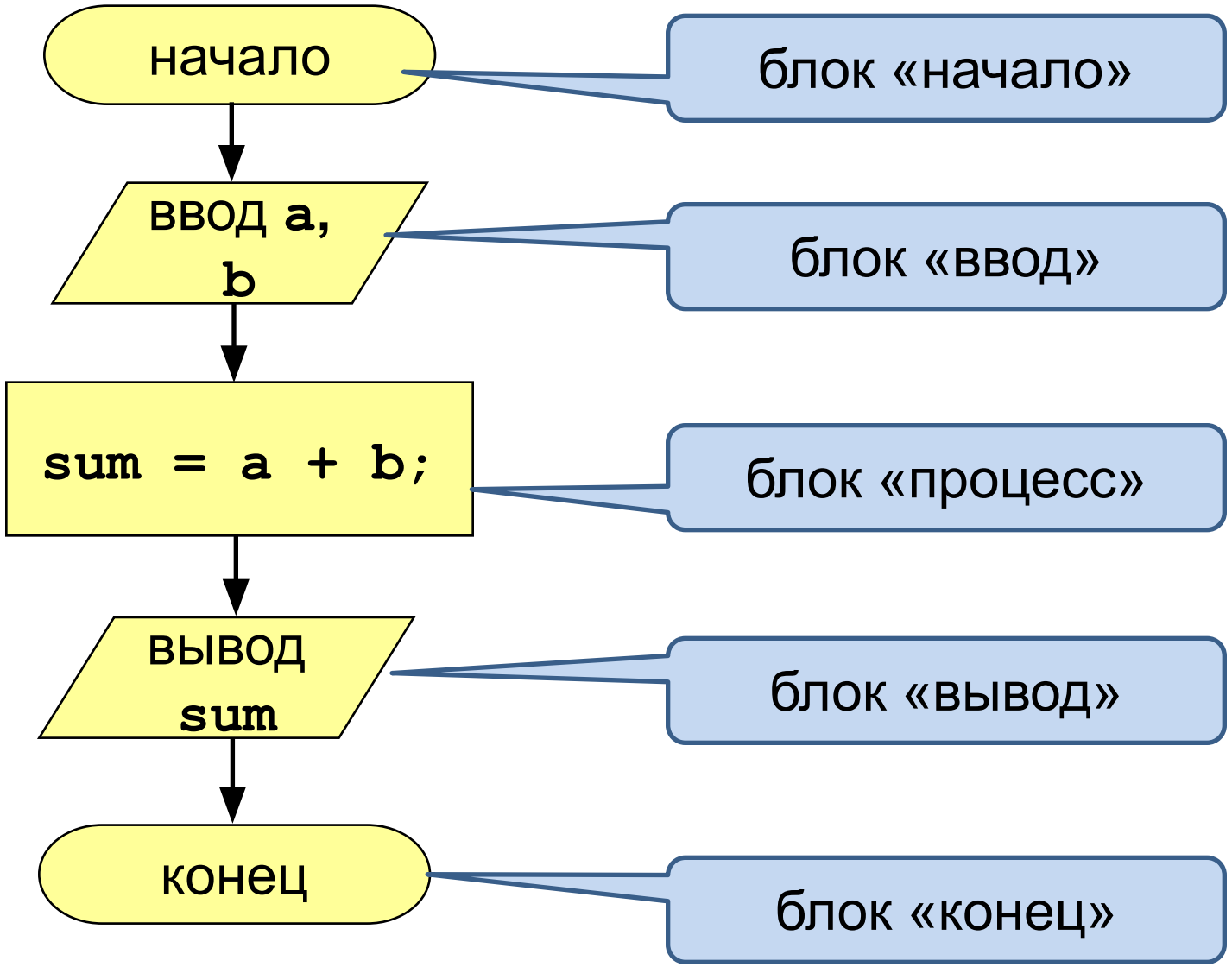
Линейный алгоритм - это алгоритм, в котором команды выполняются последовательно одна за другой.

Задача: составить программу вычисления суммы двух целых чисел.

Алгоритм.

1. Ввести два целых числа.
2. Вычислит сумму.
3. Вывести на экран результат.

Блок-схема линейного алгоритма



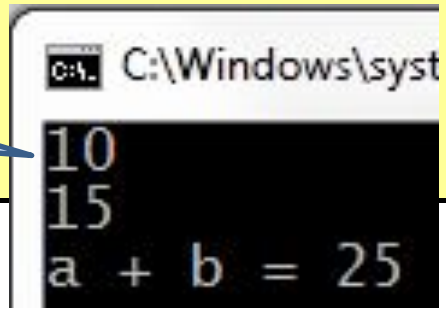
Сложение двух чисел

Простейшее решение:

```
#include <cstdio>
int main() {
   int a,b;
   scanf("%d%d",&a, &b);
   int sum;
   sum = a + b;
   printf("a + b = %d\n", sum);
   return 0;
}
```

? Что плохо?

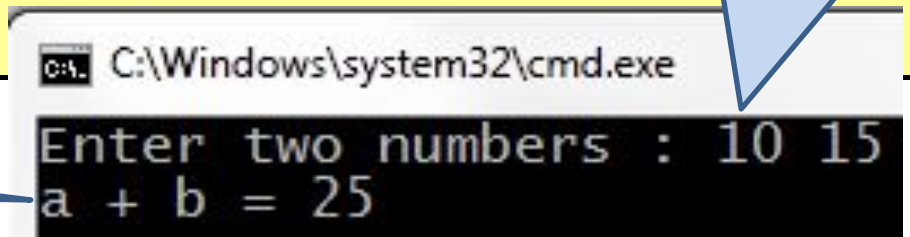
пользователь



Полное решение

```
#include <cstdio> // <stdio.h>
int main() {
   printf("Enter two numbers : ");
   int a,b;
   scanf("%d%d",&a, &b);
   int sum;
   sum = a + b;
   printf("a + b = %d\n", sum);
   return 0;
}
```

пользователь

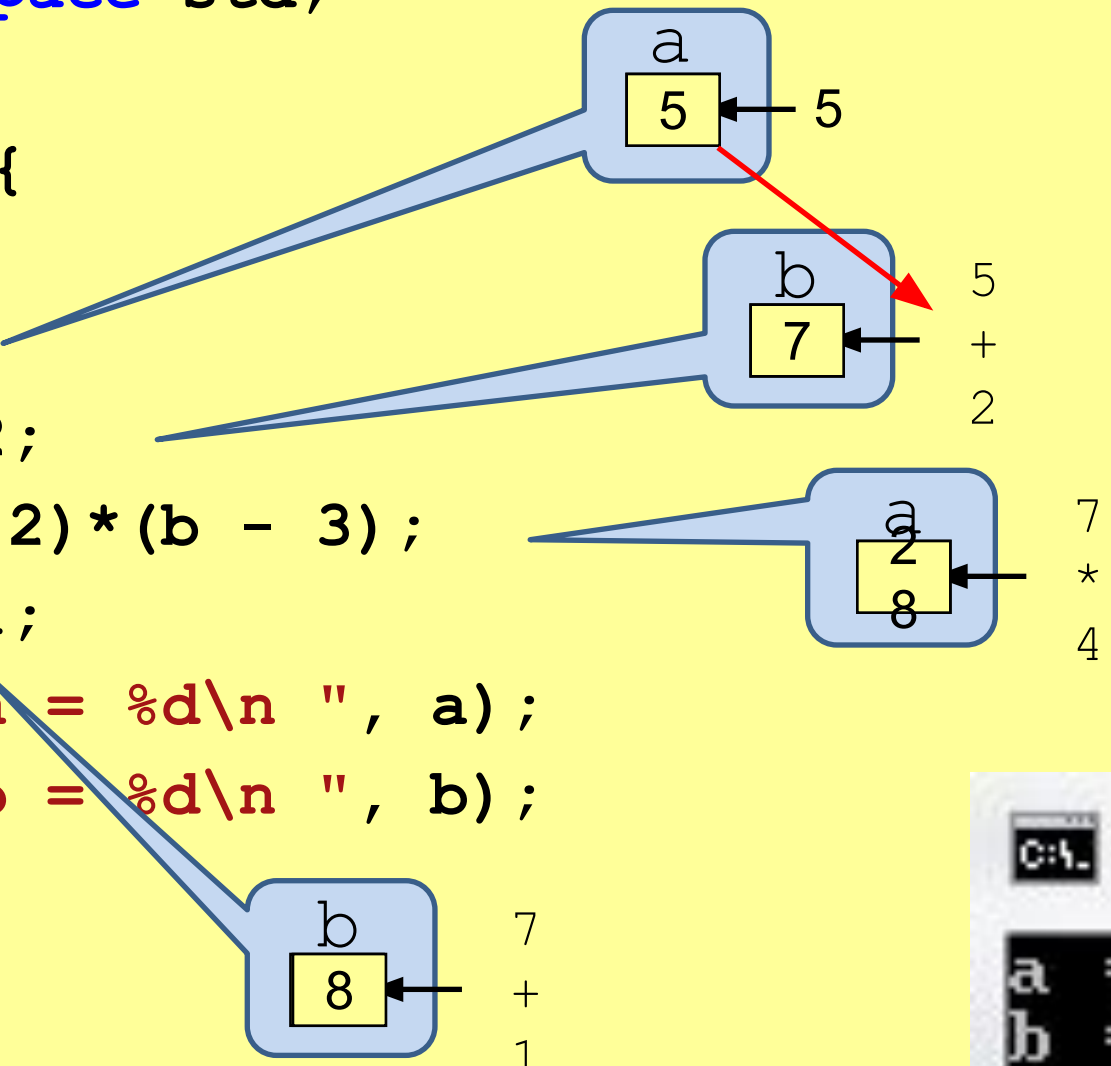


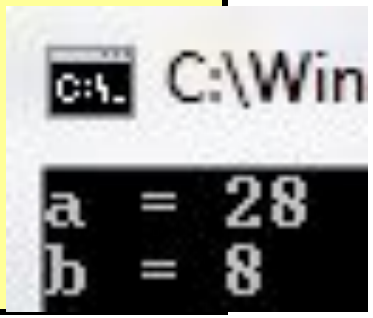
компьютер

Как изменить значение

```
#include <cstdio>
using namespace std;

int main() {
   int a, b;
   a = 5;
   b = a + 2;
   a = (a + 2)*(b - 3);
   b = b + 1;
   printf("a = %d\n ", a);
   printf("b = %d\n ", b);
   return 0;
}
```





Уметь в уме выполнять программу!
БЕЗ КОМПЬЮТЕРА
Уметь читать чужие программы

Арифметические операции

`+` сложение `-` вычитание

`*` умножение `/` деление

`%` остаток от деления

```
int a, b, x, y;
a = 7*3 - 4;
a = a * 5;
b = a / 10;
a = a % 10;
x = b % 2;
y = a % 2;
```

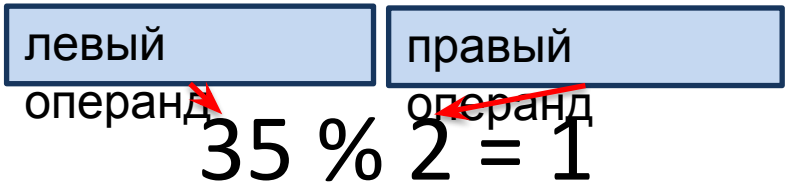
Задание! Выполните действия фрагмента программы

Остаток от деления

Операция остаток от деления применяется только к целым числам **типа** `char`, `short`, `int` и `long` и обозначается знаком **процента** `%`.
Результат операции - **остаток**, получаемый при делении левого операнда на правый.

Математика 35 : 2 = 17 (**1 ост**) 38 : 2 = 19 (**0 ост**)
94 : 10 = 9 (**4 ост**) 27 : 3 = 9 (**0 ост**) 35 : 3 = 11(**2ост**)

Программирование



По остатку можно сказать: «**число чётное**», «**число нечётное**», «**число кратно 3**» - это значит делится на 3 без остатка, «**число некратно 3**» - это значит делится на 3 с остатком

Остаток от деления

```
// применение операции остатка от деления
#include <cstdio>

int main() {
   printf("%d\n", 6 % 8);
   printf("%d\n", 7 % 8);
   printf("%d\n", 8 % 8);
   printf("%d\n", 9 % 8);
   printf("%d\n", 10 % 8);
   return 0;
}
```

Задание! Выполните действия программы.

Арифметические операции

```
#include <cstdio>
int main() {
    int a, b;
    a = 7*3 - 4;
    printf("a = %d\n ", a);
    a = a * 5;
    printf("a = %d\n ", a);
    b = a / 10;
    printf("a = %d\n ", b);
    a = a % 10;
    printf("a = %d\n ", a);
    float x = 12.0, y = 2.5;
    printf("x = %f  y = %f\n", x, y);
    float z = x / y;
    printf(""z = %f\n" , z);
    return 0;
}
```

Задание! Выполните действия программы.

Какие операторы неправильные?

```
#include <cstdio>

int main() {
   int a, b;
   float x , y;
      a = 5;
      10 = x;
      y = 7,8;
      b = 2.5;
      x = 2*(a + y);
      a = b + x;
return 0;
}
```

имя переменной должно быть слева от знака =

целая и дробная часть отделяются точкой

Чему равно b?

Чему равно a?

Уметь искать ошибки

Правильно

```
#include <cstdio>
using namespace std;

int main() {
   int a, b;
   float x , y;
   a = 5; // инициализация переменной
      x =10; // инициализация переменной
      y = 7.8; // инициализация переменной
      b = 2; // инициализация переменной
      x = 2*(a + y);
      a = b + x;
   printf("b = %d\t a = %d\n", b, a);
   printf("x = %f\t y = %f\n",x, y);
      return 0;
}
```

Задание! Выполните действия программы

Арифметические операции с присваиванием

Арифметические операции с присваиванием сокращают размер кода и делают его наглядным.

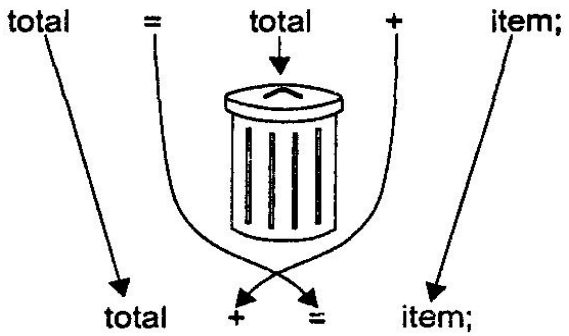
+= и **-=** сложение и вычитание с присваиванием

***=** и **/=** умножение и деление с присваиванием

%= остаток от деления с присваиванием

Например, total = total + item; // сложение total и item

```
total += item;  // сложение total и item
```



Арифметические операции с

```
// применение арифметических операций с присваиванием
#include <cstdio>
using namespace std;

int main() {
    int s = 27;
    s += 10; // то же самое, что s = s + 10;
    printf("%d, " ,s);
    s -= 7;  // то же самое, что s = s - 7;
    printf("%d, " ,s);
    s *= 2;  // то же самое, что s = s * 2;
    printf("%d, " ,s);
    s /= 3;  // то же самое, что s = s / 3;
    printf("%d, " ,s);
    s %= 3;  // то же самое, что s = s % 3;
    printf("%d\n" ,s);
    return 0;
}
```

Задание! Выполните действия программы.

Операция инкремент ++

Инкремент – это **увеличение** величины **на единицу**

```
count = count + 1; // увеличение count на 1

count += 1;   // увеличение count на 1

++count; // увеличение count на 1
count++; // увеличение count на 1
```

Операция ++ инкрементирует или увеличивает операнд **на 1**.

Операция инкремент ++

Например :

c = 10; b = 2;

a = b * ++ c;

d = b * c ++;

Если используем префиксную форму записи инкремента (**++ c**), то в этом случае инкремент будет выполнен первым, а затем уже умножение. В результате получим значение переменной **a** = **22** и значение переменной **c** = **11**.

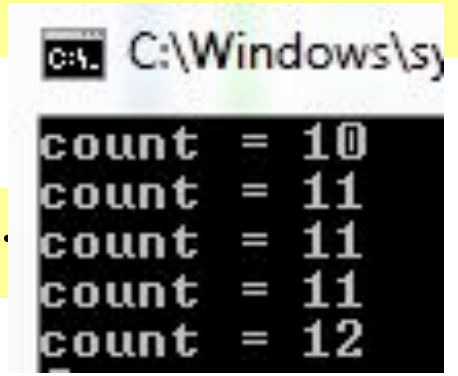
Если используем постфиксную форму записи инкремента (**c ++**), то в этом случае сначала будет выполнено умножение, а затем значение переменной с увеличится на 1.

В результате получим значение переменной **d** = **20** и

Инкремент ++

```
// применение операции инкрементирования
#include <iostream>

int main() {
    int count = 10;
    printf("count = %d\n", count); // вывод числа 10
    printf("count = %d\n", ++count); // вывод
//числа 11 (префиксная форма)
    printf("count = %d\n", count); // вывод числа 11
    printf("count = %d\n", count++); // вывод
//числа 11 (постфиксная форма)
    printf("count = %d\n", count); // вывод числа 12
    return 0;
}
```



Задание! Отличие `count++` и `++coun`

Операция декремент --

Декремент – это **уменьшение** величины **на единицу**

```
count = count - 1; // уменьшение count на 1

count -= 1;    // уменьшение count на 1

--count;   // уменьшение count на 1
count--;   // уменьшение count на 1
```

Операция **--** уменьшает операнд **на 1**.

Операция декремент - -

Результатом этой операции является уменьшение значения операнда на 1. Эта операция также может быть записана как в префиксной форме так и в постфиксной форме записи.

Например :

```
c = 10; b = 2;
a = b * -- c;        // a = 18, c = 9
d =  b * c --;       // d = 20  c = 9.
```

Выражения

Математика

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

$$z = \frac{5a+c}{ab}(b-c)$$

Программирование

```
x =(5*c*c-d*(a+b))/((c+d)*(d-2*a));
```

```
z = (5*a+c)*(b-c)/(a*b);
```

Уметь правильно записывать выражение

Порядок выполнения операций

1) вычисление выражений в скобках
2) умножение, деление, % слева направо
3) сложение и вычитание слева направо

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

$$z = \frac{5a + c}{ab}(b - c)$$

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

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

Чему равны a и b?

```
#include <cstdio>

int main() {
   int a, b;
   a = 5;
   b = a + 2;
   a = (a + 2)*(b - 3);
   b = a / 5;
   a = a % b;
      a = a + 1;
      b = (a + 14) % 7;
   return 0;
}
```

a	b
?	?

Задание! Выполните действия программы.

Задание. Напишите программы

1. Ввести три числа, найти их сумму и произведение.

   Пример:

```
Введите три числа:
4   5   7
4+5+7=16
4*5*7=140
```

2. Ввести три числа, найти их сумму, произведение и среднее арифметическое.

   Пример:

```
    Введите три числа:
4   5   7
4+5+7=16
4*5*7=140
(4+5+7)/3=5.33
```