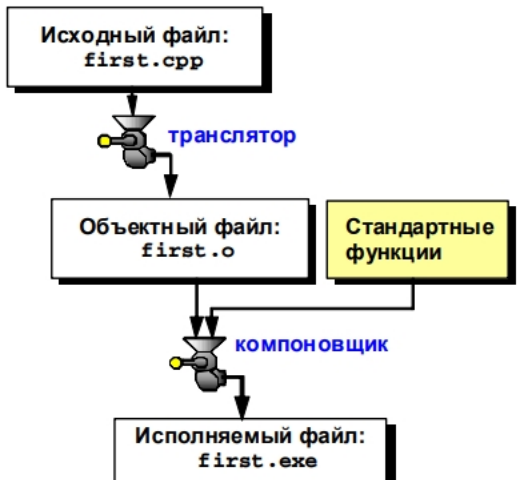
Два этапа создания программ

Программа на языке Си, так же как и на большинстве современных языков программирования, создается в два этапа

1) **трансляция** – перевод текста программы в машинные коды;
2) **компоновка** – сборка частей программы и подключение стандартных функций.



Вывод текста на экран

Составим теперь программу, которая делает что-нибудь полезное, например, выводит на экран слово «Привет».

```
#include <stdio.h>
main()
{
printf("Привет");
}
```

подключение функций стандартного ввода и вывода, описание которых находится в файле **stdio.h**

вызов функции вывода на экран

Остановим мгновение

Если запускать рассмотренную выше программу, то обнаружится, что программа сразу заканчивает работу и возвращается обратно в оболочку, не дав нам посмотреть результат ее работы на экране. Бороться с этим можно так – давайте скажем компьютеру, что в конце работы надо дождаться нажатия клавиши ***Enter***, а только потом закрывать окно.

```
#include <stdio.h>
main()
{
printf("Привет");       // вывод на экран
getchar();             /*ждать нажатия клавиши Enter*/
}
```

2. Переменные

Типы данных и переменные

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.

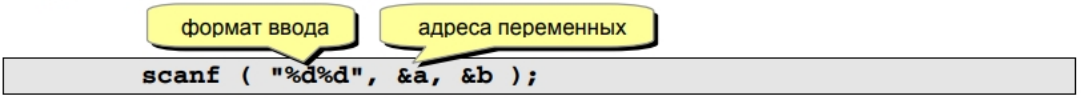
- **целые** переменные – тип **`int`** (от английского *integer* – целый), занимают 4 байта в памяти;
- **вещественные** переменные, которые могут иметь дробную часть (тип **`float`** – от английского *floating point* – плавающая точка) , занимают 4 байта в памяти
- **символы** (тип **`char`** – от английского *character* – символ), занимают 1 байт в памяти

```
int a;        // выделить память под целую переменную a

float b, c;  // две вещественных переменных b и c

int Tu104, I186=23, Yak42; // три целых переменных,
                           // в I186 сразу записывается число 23

float x=4.56, y, z;     // три вещественных переменных,
                        // в x сразу записывается число 4.56

char c, c2='A', m;      // три символьных переменных, в c2
                        // сразу записывается символ 'A'
```

Вычисление суммы двух чисел (ввод и вывод)

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

```c
#include <stdio.h>
main()
{
int a, b, c;  // объявление переменных
printf ( "Введите два целых числа \n" ); // подсказка для ввода
scanf ( "%d%d", &a, &b );                // ввод данных
c = a + b;              // вычисления (оператор присваивания)
printf ( "Результат: %d + %d = %d \n",
         a, b, c );                      // вывод результата
getchar();
}
```



- Формат ввода – это строка в кавычках, в которой перечислены один или несколько форматов ввода данных:

%d	ввод целого числа (переменная типа **int**)
%f	ввод вещественного числа (переменная типа **float**)
%c	ввод одного символа (переменная типа **char**)

- После формата ввода через запятую перечисляются адреса ячеек памяти, в которые надо записать введенные значения. Почувствуйте разницу:

a	значение переменной **a**
&a	адрес переменной **a**

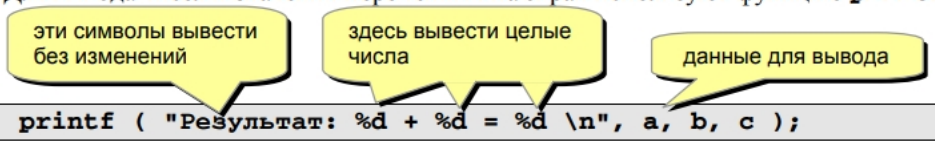
- Количество форматов в строке должно быть равно количеству адресов переменных в списке. Кроме того, тип переменных должен совпадать с указанным: например, если **a** и **b** – целые переменные, то следующие вызовы функций ошибочны

`scanf ( "%d%d", &a );`	куда записывать второе введенное число?
`scanf ( "%d%d", &a, &b, &c );`	не задан формат для переменной **c**
`scanf ( "%f%f", &a, &b );`	нельзя вводить целые переменные по вещественному формату

- Для вычислений используют **оператор присваивания**, в котором
  - справа от знака равенства стоит арифметическое выражение, которое надо вычислить
  - слева от знака равенства ставится имя переменной, в которую надо записать результат

```
c = a + b;        // сумму a и b записать в c
```

- Для вывода чисел и значений переменных на экран используют функцию **printf**



содержание скобок при вызове функции **printf** очень похоже на функцию **scanf**

- Сначала идет символьная строка — формат вывода — в которой можно использовать специальные символы
  - **%d** вывод целого числа
  - **%f** вывод вещественного числа
  - **%c** вывод одного символа
  - **%s** вывод символьной строки
  - **\n** переход в начало новой строки

  все остальные символы (кроме некоторых других специальных команд) просто выводятся на экран.

- Одной строки формата недостаточно: в ней сказано, в какое место выводить данные, но не сказано, откуда их взять. Поэтому через запятую после формата вывода надо поставить список чисел или переменных, значения которых надо вывести, при этом можно сразу проводить вычисления.

```
printf ( "Результат: %d + %d = %d \n", a, 5, a+5 );
```

- Так же, как и для функции **scanf**, надо следить за совпадением типов и количества переменных и форматов вывода.

Арифметические выражения

Из чего состоят арифметические выражения?

Арифметические выражения, стоящие в правой части оператора присваивания, могут содержать

- целые и вещественные числа (в вещественных числах целая и дробная часть разделяются **точкой**, а не запятой, как это принято в математике)
- знаки арифметических действий

`+ –`	сложение, вычитание
`* /`	умножение, деление
`%`	остаток от деления

- вызовы стандартных функций

`abs(i)`	модуль целого числа `i`
`fabs(x)`	модуль вещественного числа `x`
`sqrt(x)`	квадратный корень из вещественного числа `x`
`pow(x,y)`	вычисляет `x` в степени `y`

- круглые скобки для изменения порядка действий

Особенности арифметических операций

При использовании деления надо помнить, что

При делении целого числа на целое остаток от деления отбрасывается, таким образом, 7/4 будет равно 1. Если же надо получить вещественное число и не отбрасывать остаток, делимое или делитель надо преобразовать к вещественной форме. Например:

```
int i, n;
float x;
i = 7;
x = i / 4;          // x=1, делится целое на целое
x = i / 4.;         // x=1.75, делится целое на дробное
x =(float) i / 4; // x=1.75, делится дробное на целое
n = 7. / 4.;        // n=1, результат записывается в
                    // целую переменную
```

Наибольшие сложности из всех действий вызывает взятие остатка. Если надо вычислить остаток от деления переменной **a** на переменную **b** и результат записать в переменную **ostatok**, то оператор присваивания выглядит так:

```
   ostatok = a % b;
```

Приоритет арифметических операций

В языках программирования арифметические выражения записываются в одну строчку, поэтому необходимо знать **приоритет** (старшинство) операций, то есть последовательность их выполнения. Сначала выполняются

- операции в скобках, затем…
- вызовы функций, затем…
- умножение, деление и остаток от деления, слева направо, затем…
- сложение и вычитание, слева направо.

3. Выбор вариантов

Зачем нужны ветвления?

В простейших программах все команды выполняются одна за другой последовательно. Так реализуются *линейные* алгоритмы. Однако часто надо выбрать тот или иной вариант действий в зависимости от некоторых условий: если условие верно, поступать одним способом, а если неверно — другим. Для этого используют **разветвляющиеся алгоритмы**, которые в языках программирования представлены в виде **условных операторов**. В языке Си существует два вида условных операторов:

- оператор **`if – else`** для выбора из двух вариантов
- оператор множественного выбора **`switch`** для выбора из нескольких вариантов

Условный оператор if – else

Задача. Ввести с клавиатуры два вещественных числа и определить наибольшее из них.

По условию задачи нам надо вывести один из двух вариантов ответа: если первое число больше второго, то вывести на экран его, если нет — то второе число. Ниже показаны два варианта решения этой задачи: в первом результат сразу выводится на экран, а во втором наибольшее из двух чисел сначала записывается в третью переменную **Max**.

```
#include <stdio.h>
main()
{
float A, B;
printf ("Введите A и B :");
scanf ( "%f%f", &A, &B );
if ( A > B )
 {
 printf ( "Наибольшее %f",
           A );
 }
else
 {
 printf ( "Наибольшее %f",
           B );
 }
getchar();
}
```

```
#include <stdio.h>
main()
{
float A, B, Max;
printf("Введите A и B : ");
scanf ( "%f%f", &A, &B );
if ( A > B ) // заголовок
     {
     Max = A; // блок «если»
     }
else
     {
     Max = B; // блок «иначе»
     }
printf ( "Наибольшее %f",
           Max );
getchar();
}
```

- Условный оператор имеет следующий вид:

```
if ( условие ) // заголовок с условием
   {
   ... // блок «если» – операторы, которые выполняются,
       // если условие в заголовке истинно
   }
else
  {
```

```
    ...    // блок «иначе» – операторы, которые выполняются,
           // если условие в скобках ложно
  }
```

- Эта запись представляет собой единый оператор, поэтому между скобкой, завершающей блок «если» и словом **else** не могут находиться никакие операторы.
- После слова **else** никогда **НЕ** ставится условие — блок «иначе» выполняется тогда, когда основное условие, указанное в скобках после **if**, ложно.
- Если в блоке «если» или в блоке «иначе» только один оператор, то фигурные скобки можно не ставить.
- В условии можно использовать знаки логических отношений

> <	больше, меньше
>= <=	больше или равно, меньше или равно
==	равно
!=	не равно

- В языке Си любое число, не равное нулю, обозначает истинное условие, а ноль — ложное условие.
- Если в блоке «иначе» не надо ничего делать (например: «если в продаже есть мороженое, купи мороженое», а если нет ...), то весь блок «иначе» можно опустить и использовать сокращенную форму условного оператора:

```
if ( условие )
   {
   ...    // что делать, если условие истинно
   }
```

Сложные условия

Простейшие условия состоят из одного отношения (больше, меньше и т.д.). Иногда надо написать условие, в котором объединяются два или более простейших отношений. Например, фирма отбирает сотрудников в возрасте от 25 до 40 лет (включительно). Тогда простейшая программа могла бы выглядеть так:

```
#include <stdio.h>
main()
{
int age;
printf ( "\nВведите ваш возраст: " );
scanf ( "%d", &age );
if ( 25 <= age  &&  age <= 40 )  // сложное условие
   printf ( "Вы нам подходите." );
else
   printf ( "Извините, Вы нам не подходите." );
getchar();
}
```

Переключатель switch (множественный выбор)

Если надо выбрать один из нескольких вариантов в зависимости от значения некоторой целой или символьной переменной, можно использовать несколько вложенных операторов **if**, но значительно удобнее использовать специальный оператор **switch**.

Задача. Составить программу, которая вводит с клавиатуры русскую букву и выводит на экран название животного на эту букву.

```
#include <stdio.h>
main()
{
char c;
printf("\nВведите первую букву:");
scanf("%c", &c);  // ввести букву
switch ( c )      // заголовок оператора выбора
  {
  case 'а': printf("\nАнтилопа"); break;
  case 'б': printf("\nБарсук"); break;
  case 'в': printf("\nВолк"); break;
  default:  printf("\nНе знаю я таких!"); // по умолчанию
  }
getchar();
}
```