# Основы алгоритмизации и быстрое введение в язык Си

Алтайский государственный университет
Факультет математики и ИТ
Кафедра информатики
Барнаул 2014

# Лекция 1

- План
  - Основы алгоритмизации
  - Быстрое введение в язык Си

# Основы алгоритмизации

- Алгоритм: определение
- Алгоритм: свойства
- Алгоритм: исполнитель
- Алгоритм: формы записи
- Алгоритмические структуры
- Программы: виды ошибок
- Языки программирования

# Алгоритм: определения

- Алгоритм — заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов

- Алгоритм — точное описание последовательности действий, направленных на получение из заданного начального состояния определенного результата

- Алгоритм —
  1) план того, что должно быть сделано, выполнено
  2) закодированная информация, вводимая в компьютер для управления его деятельностью

# Алгоритм: исполнитель

- **Исполнитель алгоритма** — это некоторая абстрактная или реальная (техническая, биологическая или биотехническая) система, способная выполнить действия, предписываемые алгоритмом

- Исполнителя характеризуют:
  - среда
  - элементарные действия
  - система команд исполнителя (СКИ)
  - отказы

# Алгоритм: свойства

- **Понятность** для исполнителя
  - исполнитель алгоритма должен понимать, как его выполнять
- **Дискретность** (прерывность, раздельность)
  - алгоритм должен состоять из отдельных шагов (этапов).
- **Результативность** (конечность)
  - алгоритм должен приводить к результату за конечное число шагов
- **Определенность** (детерминированность)
  - при одинаковых исходных данных алгоритм должен выдавать один и тот же результат
- **Массовость**
  - алгоритм должен решать целый класс однотипных задач с различными конкретными значениями исходных данных
- **Корректность**
  - алгоритм должен давать верное решение при любых допустимых исходных данных

# Алгоритм: формы записи

- **Словесная**
  - запись на естественном языке
- **Графическая**
  - изображения из графических символов
- **Псевдокоды**
  - полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.
- **Программная**
  - тексты на языках программирования

# Алгоритм: словесная форма записи

- Алгоритм записывается в виде пронумерованной последовательности шагов на естественном языке
- Алгоритмы в словесной форме записи адресованы человеку
- Пример:

Алгоритм сложения двух чисел (*a* и *b*)

1. Спросить, чему равно число *a*
2. Спросить, чему равно число *b*
3. Сложить *a* и *b*, результат присвоить *c*
4. Сообщить результат *c*

# Алгоритм: графическая форма записи

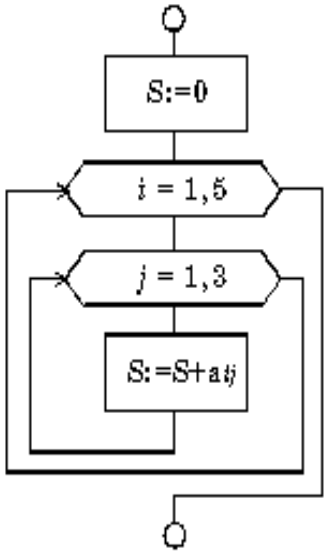
- При графическом представлении алгоритм изображается в виде **последовательности связанных** между собой функциональных **блоков**, каждый из которых соответствует выполнению одного или нескольких действий

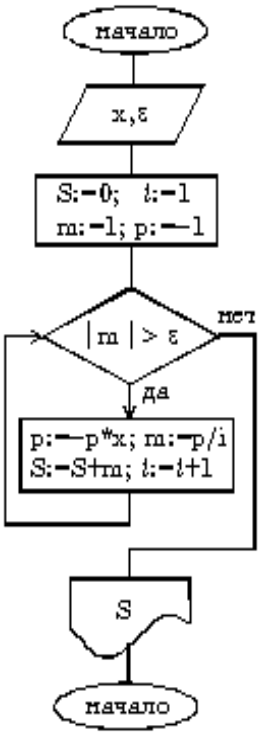
- Алгоритмы в графической форме записи адресованы преимущественно человеку

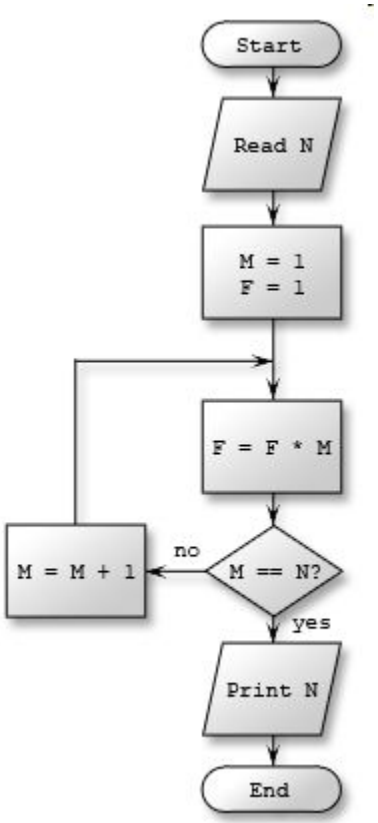
- Примеры:
  - Схема сборки мебели
  - Блок-схемы алгоритмов

# Алгоритм: графическая форма записи

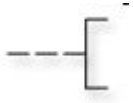
- Блок-схемы алгоритмов
  - См. http://ru.wikipedia.org/wiki/Блок-схема



# Блок-схемы: основные обозначения

| | | |
|---|---|---|
| **Терминатор** |  | Отображает вход или выход из внешней среды (чаще всего начало и конец программы). |
| **Процесс** |  | Выполнение операций, обработка данных. Внутри указываются операции. Например: *a* = 10**b* + *c*. |
| **Решение** |  | Отображает функцию переключательного типа с одним входом и двумя или более альтернативными выходами. |
| **Предопределенный процесс** |  | Отображает выполнение именованного процесса, определенного в другом месте программы (подпрограмма). |
| **Ввод-вывод** |  | Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод). |
| **Границы цикла** |  | Символ из двух частей – начала и конца цикла. Операции, выполняемые внутри цикла, размещаются между ними. Условия цикла и приращения записываются внутри символа начала или конца цикла в зависимости от типа цикла. |
| **Соединитель** |  | Отображает выход в часть схемы и вход из другой части этой схемы. Используется для обрыва/продолжения линии. |
| **Комментарий** |  | Используется для более подробного описания шага, процесса или группы процессов. |

# Алгоритм: запись в псевдокодах

- Псевдокод
  - представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов;
  - занимает промежуточное место между естественным и формальным языками;
  - обычно  не подчинен строгим синтаксическим правилам записи команд;
  - содержит некоторые конструкции, присущие формальным языкам.

- Примеры:
  - Алгоритм на школьном алгоритмическом языке
  - Паскаль + русский язык

# Алгоритм: запись в псевдокодах

- Пример. Алгоритм на школьном алгоритмическом языке

```
алг Сумма квадратов (арг цел n, рез цел S)
    дано | n > 0
    надо | S = 1*1 + 2*2 + 3*3 + ... + n*n
нач цел I
    ввод n; S:=0
    нц для i от 1 до n
      S:=S+i*I
    кц
    вывод "S = ", S
кон
```

# Алгоритм: программная запись

- Программная запись
  - запись на специальном языке программирования
  - осуществляется строго в соответствии с правилами (синтаксисом) языка программирования
  - Понятна человеку, но предназначена формальному исполнителю (компьютеру, другому устройству, программе и т.п.)
- Примеры:
  - Программа на языке программирования Pascal
  - Программа на языке описания графических сцен POV-Ray
  - Описание веб-страницы на языке разметки гипертекста HTML и языке сценариев JavaScript

# Алгоритм: программная запись

- Пример. Программа на языке программирования Pascal

```
{ Программа вычисления суммы }
var
  i, n : longint;
  x, eps, sinx, sum, f, p, sign : single;
begin
 write('x = ');   readln(x);
 write('eps = '); readln(eps);
 sinx := sin(x);
 sum := x;
 sign := 1;
 n := 1;
 while ( abs(sinx - sum) > eps) do begin
    inc(n,2);
    if n>10 then begin
       writeln('Max number of iterations exceeded!');
       break;
    end;
    f := 1;
    p := 1;
    for i := 1 to n do begin
       f := f*i;
       p := p*x;
    end;
    sign := -sign;
    sum := sum + sign*p/f;
 end;
 writeln('sin(x) = ', sinx:10:8);
 writeln('sum    = ', sum:10:8);
end.
```

# Языки программирования

- **Машинно-ориентированные языки**
  - языки **низкого уровня**
  - каждая команда соответствует одной команде процессора
  - специфичны для конкретной платформы
- Языки **высокого уровня** (алгоритмические языки)
  - приближены к естественным языкам
  - понятнее и удобнее для человека
  - не зависят от конкретного компьютера

# Языки программирования

- Арифметическое выражение

$$y = \frac{3(4x^2 + 3x)}{10 - x}$$

- Запись на алгоритмическом языке

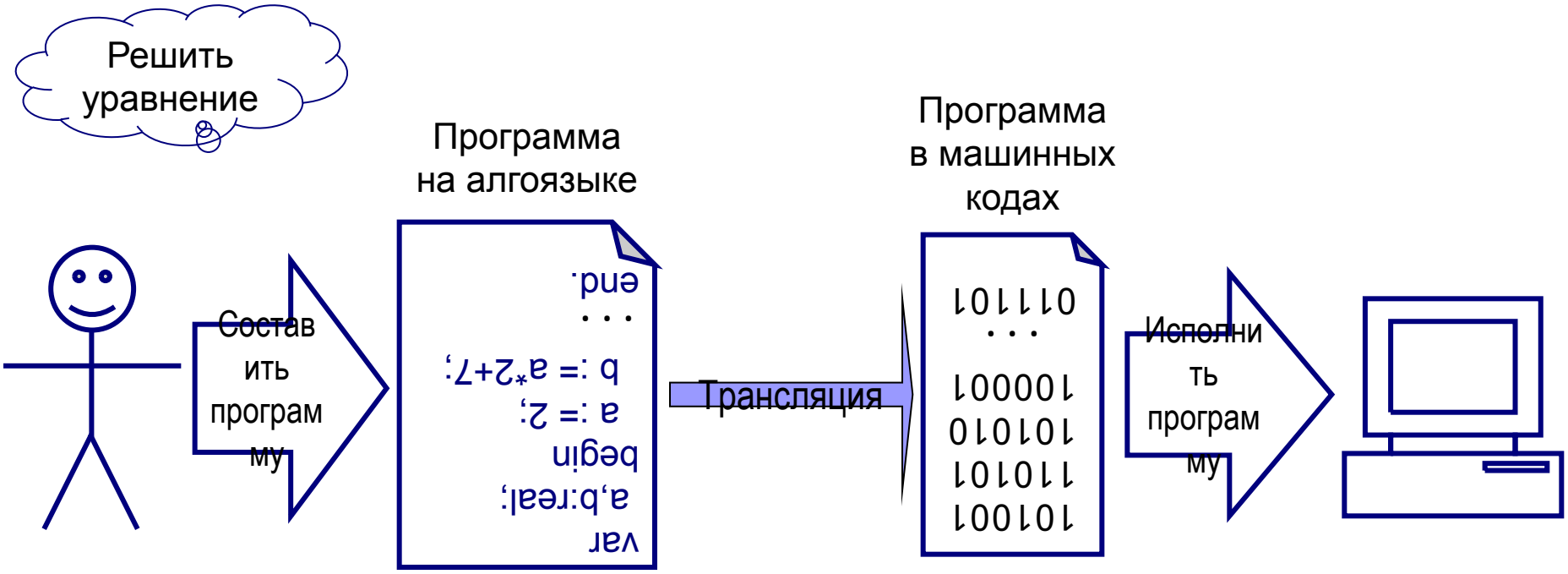
```
y = 3*(4*x*x+3*x)/(10-x);
```

- Запись на ассемблере

```
fld    dword ptr [@2]
fmul   dword ptr [ebp-4]
fmul   dword ptr [ebp-4]
fld    dword ptr [@2+4]
fmul   dword ptr [ebp-4]
faddp  st(1),st
fmul   dword ptr [@2+4]
fld    dword ptr [@2+8]
fsub   dword ptr [ebp-4]
fdivp  st(1),st
fstp   dword ptr [ebp-8]
```

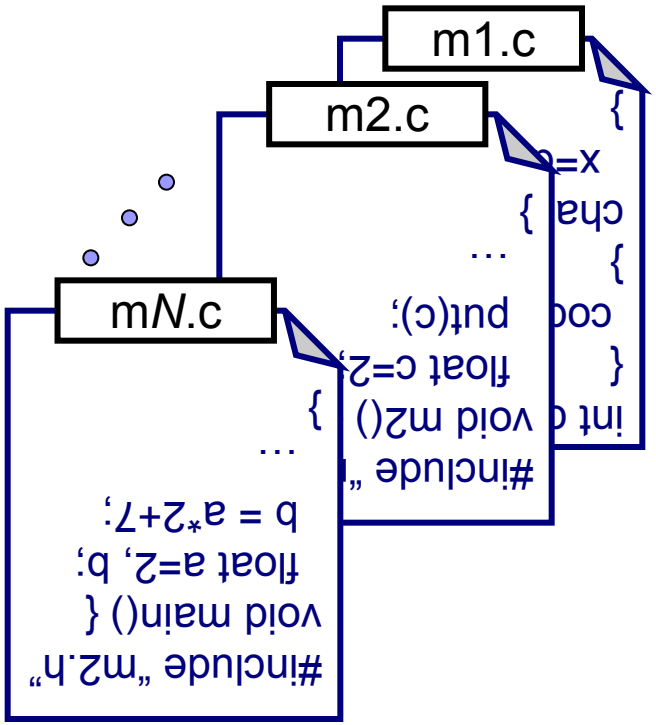
# Языки программирования

- Перевод с алгоритмического языка в числовые коды выполняет транслятор



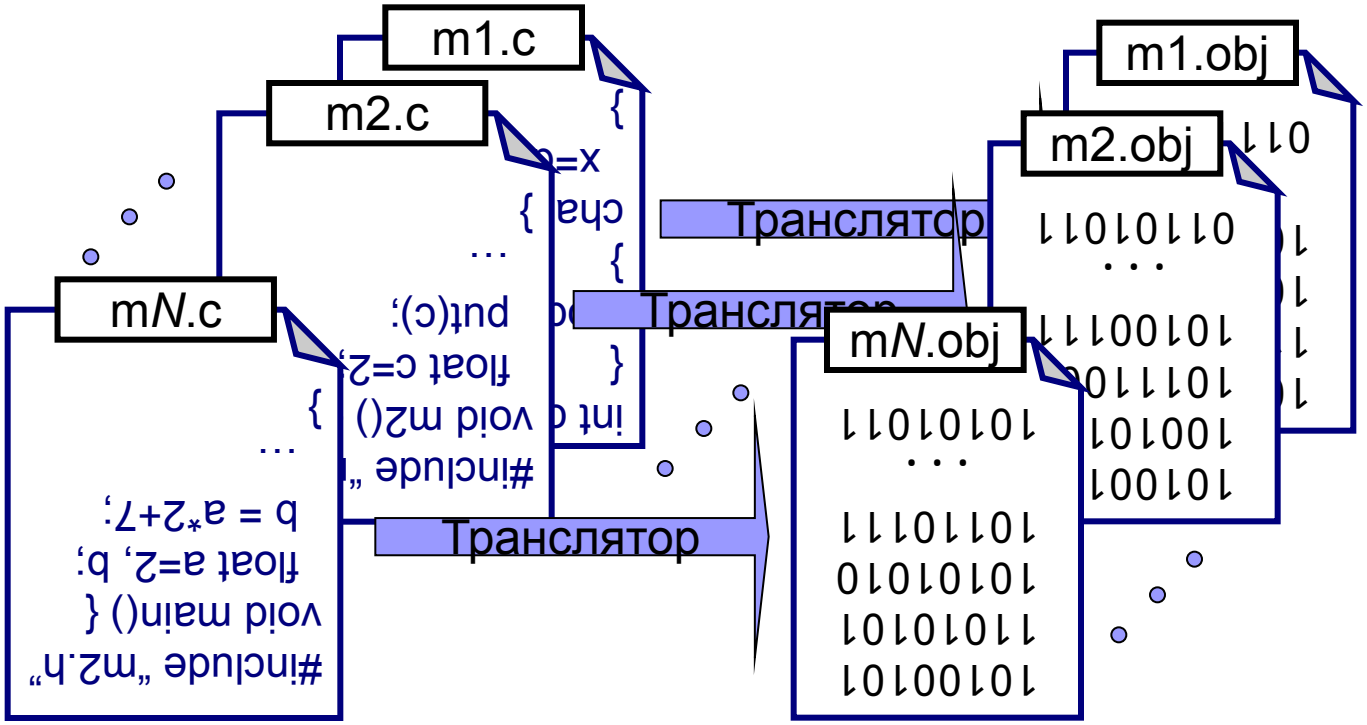
# Языки программирования

- Часто для построения программы необходимо использовать несколько разных модулей (файлов) и/или стандартные модули



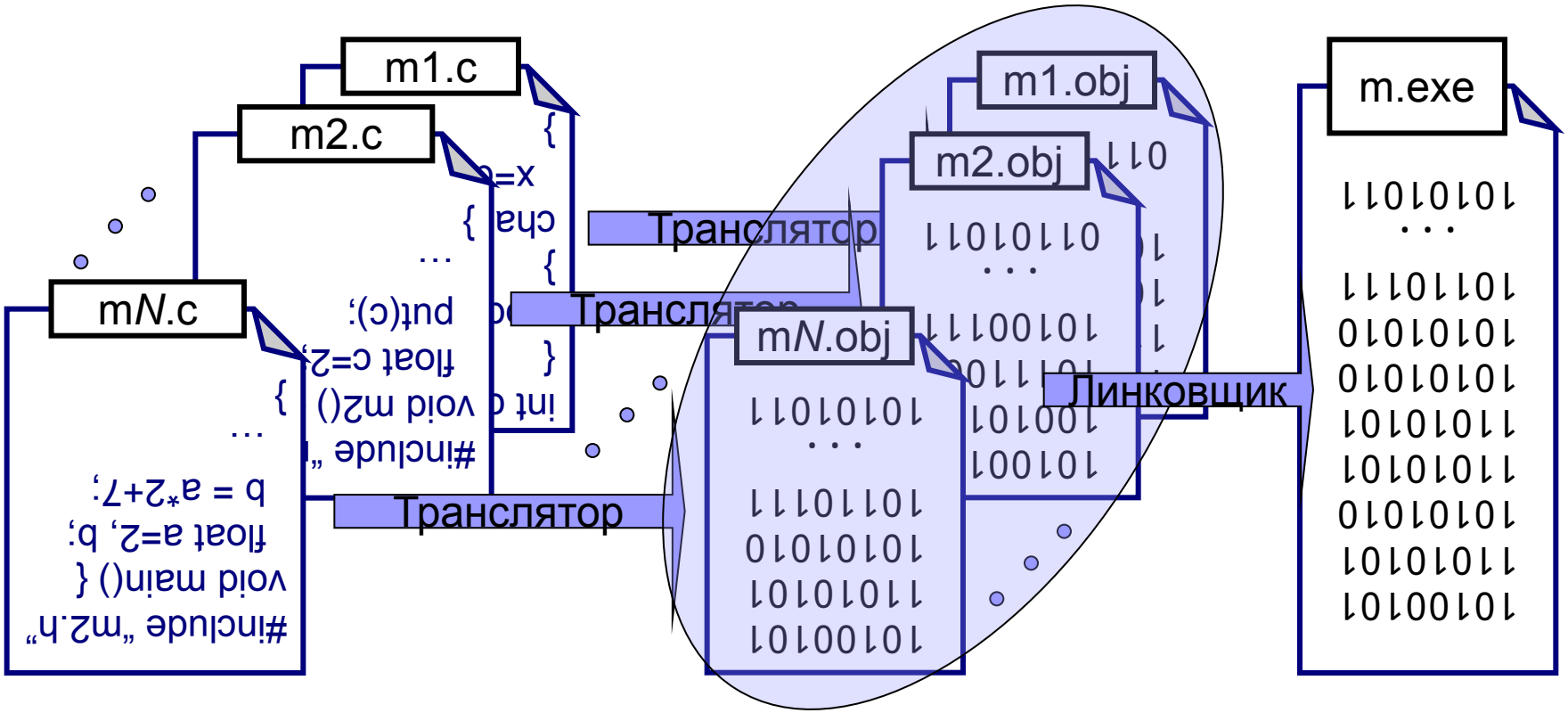
# Языки программирования

- В этом случае транслятор строит программу в числовых кодах (объектные файлы) отдельно для каждого модуля



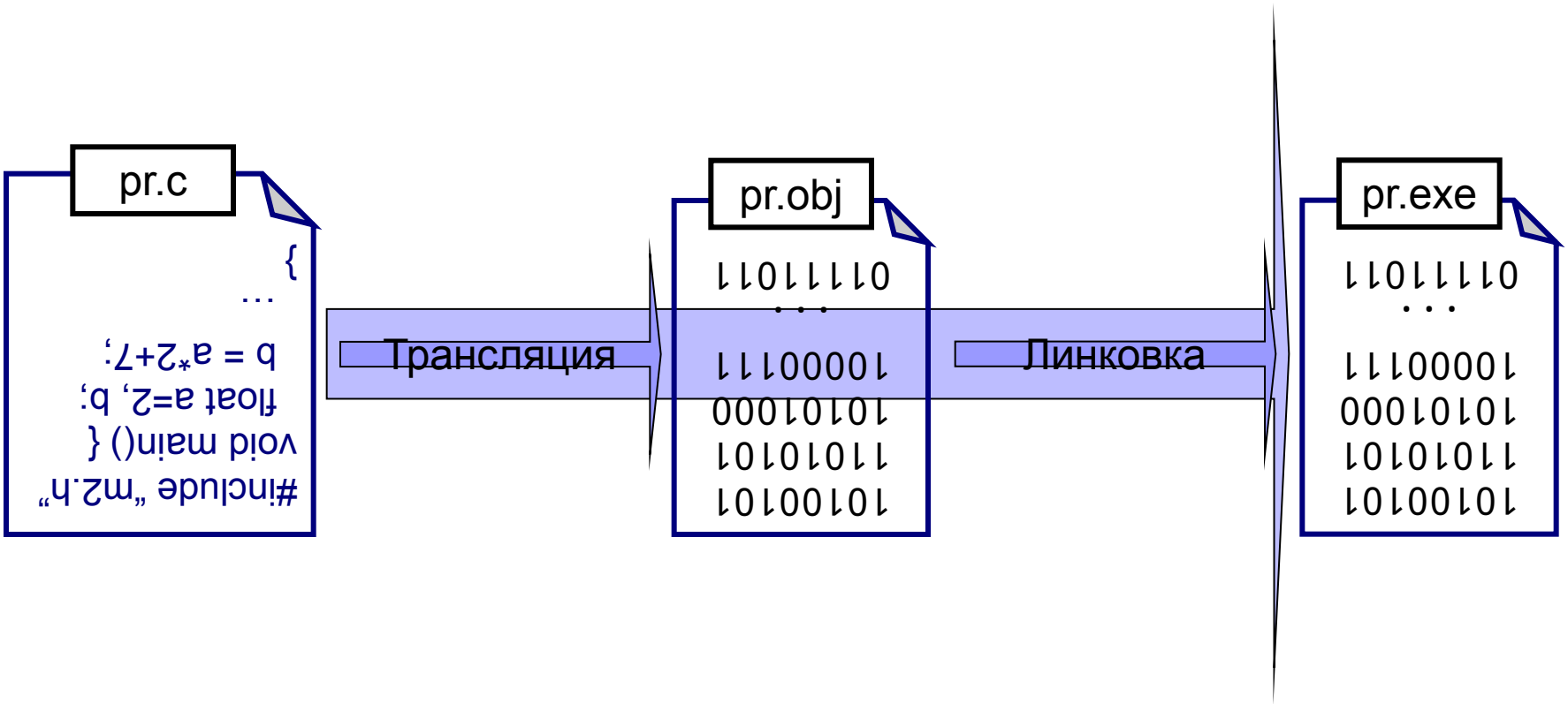
# Языки программирования

- Окончательную сборку приложения (exe-файла) из объектных файлов производит **линковщик** (*от англ.* link – связывать)



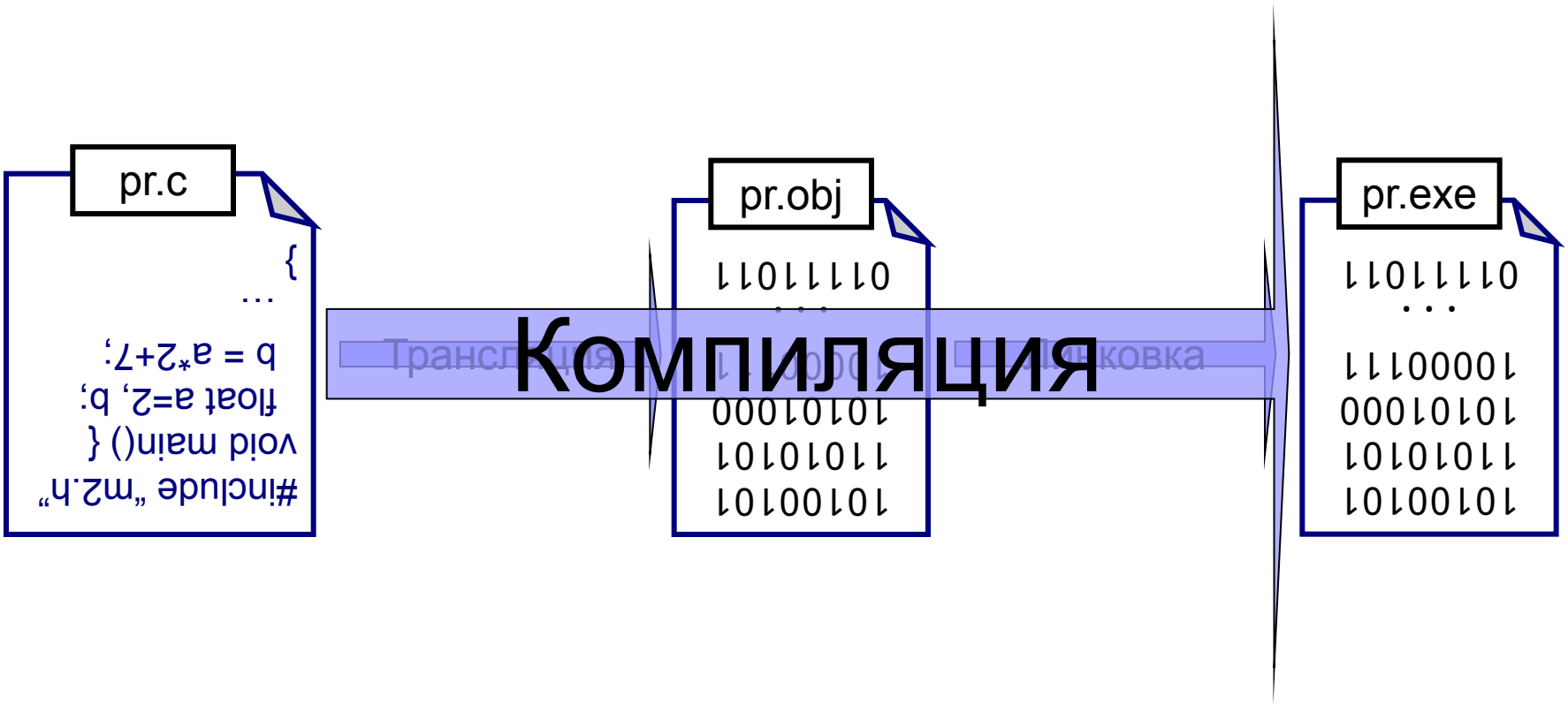
# Языки программирования

- Программа, осуществляющая и трансляцию программ, и сборку приложения из объектных файлов, называется **компилятором**



# Языки программирования

- Программа, осуществляющая и трансляцию программ, и сборку приложения из объектных файлов, называется **компилятором**



# Первичные сведения о языке Си

- Почему Си?
- История
- Простейшая программа

# Почему Си?

- Один из наиболее популярных языков программирования
- Синтаксис языка Си является основой для многих других языков программирования (C++, Java, JavaScript, C# и пр.)
- Программы на Си хорошо переносимы между различными платформами (компиляторы Си существуют, практически, для всех типов процессоров)
- Сочетает в себе черты языков низкого и высокого уровней

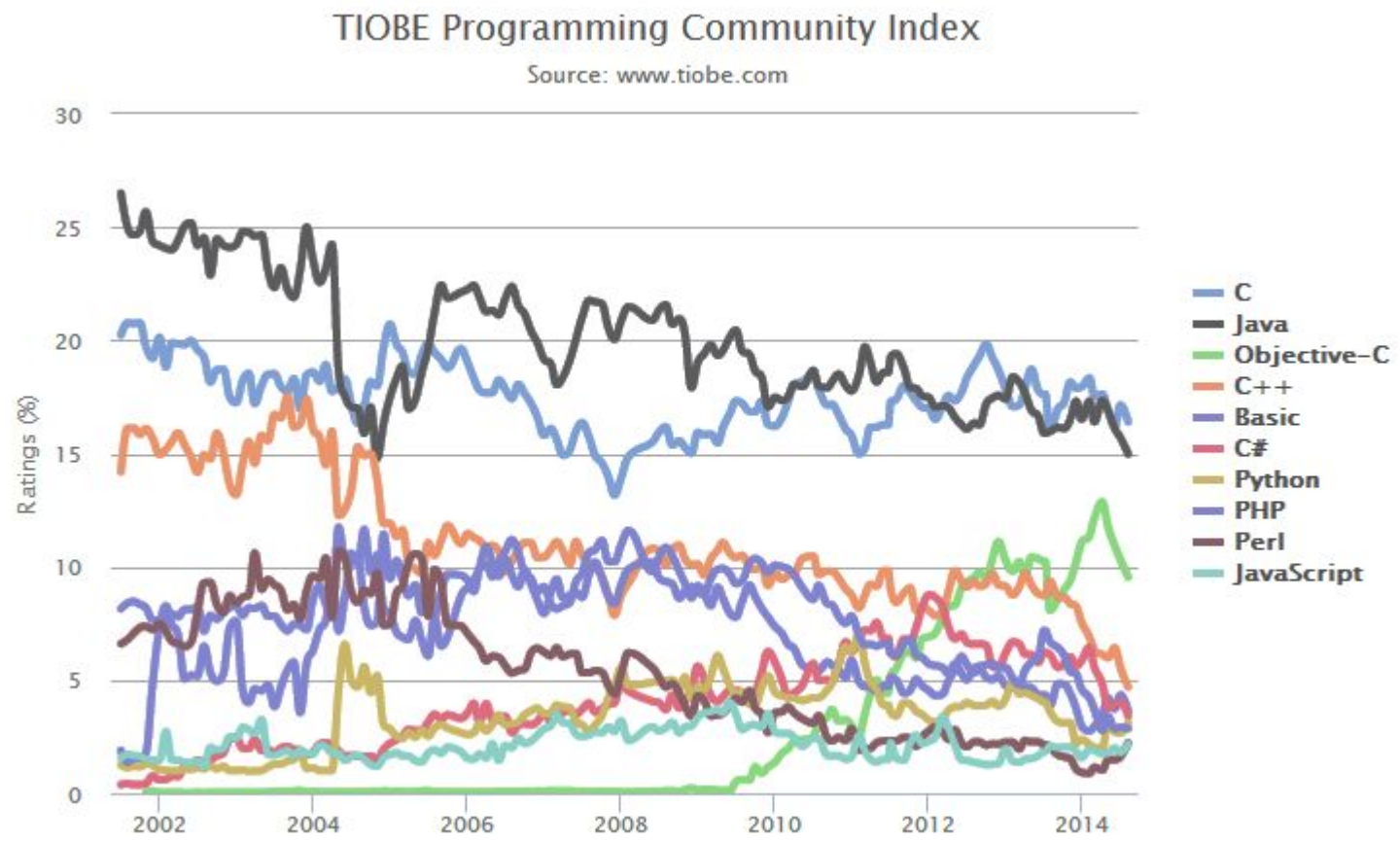
# Популярность ЯП

- TIOBE Programming Community Index for August 2014
  - Количество запросов на Google

| Aug 2014 | Aug 2013 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 2 | ^ | C | 16.401% | +0.43% |
| 2 | 1 | v | Java | 14.984% | -0.99% |
| 3 | 4 | ^ | Objective-C | 9.552% | +1.47% |
| 4 | 3 | v | C++ | 4.695% | -4.68% |
| 5 | 7 | ^ | Basic | 3.635% | -0.24% |
| 6 | 6 | | C# | 3.409% | -2.71% |
| 7 | 8 | ^ | Python | 3.121% | -0.48% |
| 8 | 5 | v | PHP | 2.864% | -3.83% |
| 9 | 11 | ^ | Perl | 2.218% | +0.18% |
| 10 | 9 | v | JavaScript | 2.172% | +0.08% |

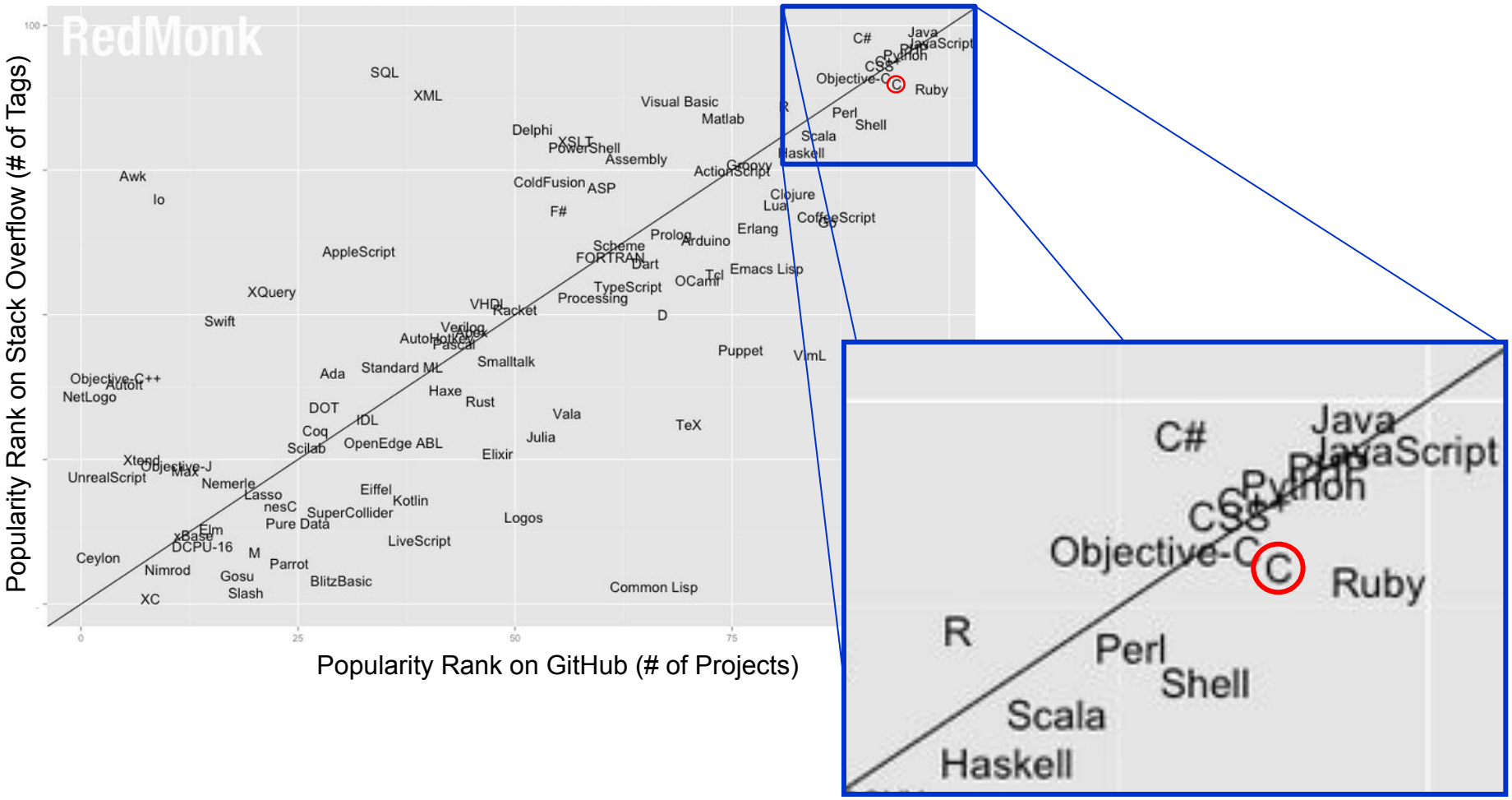
# Популярность ЯП

- TIOBE Programming Community Index for for August 2014



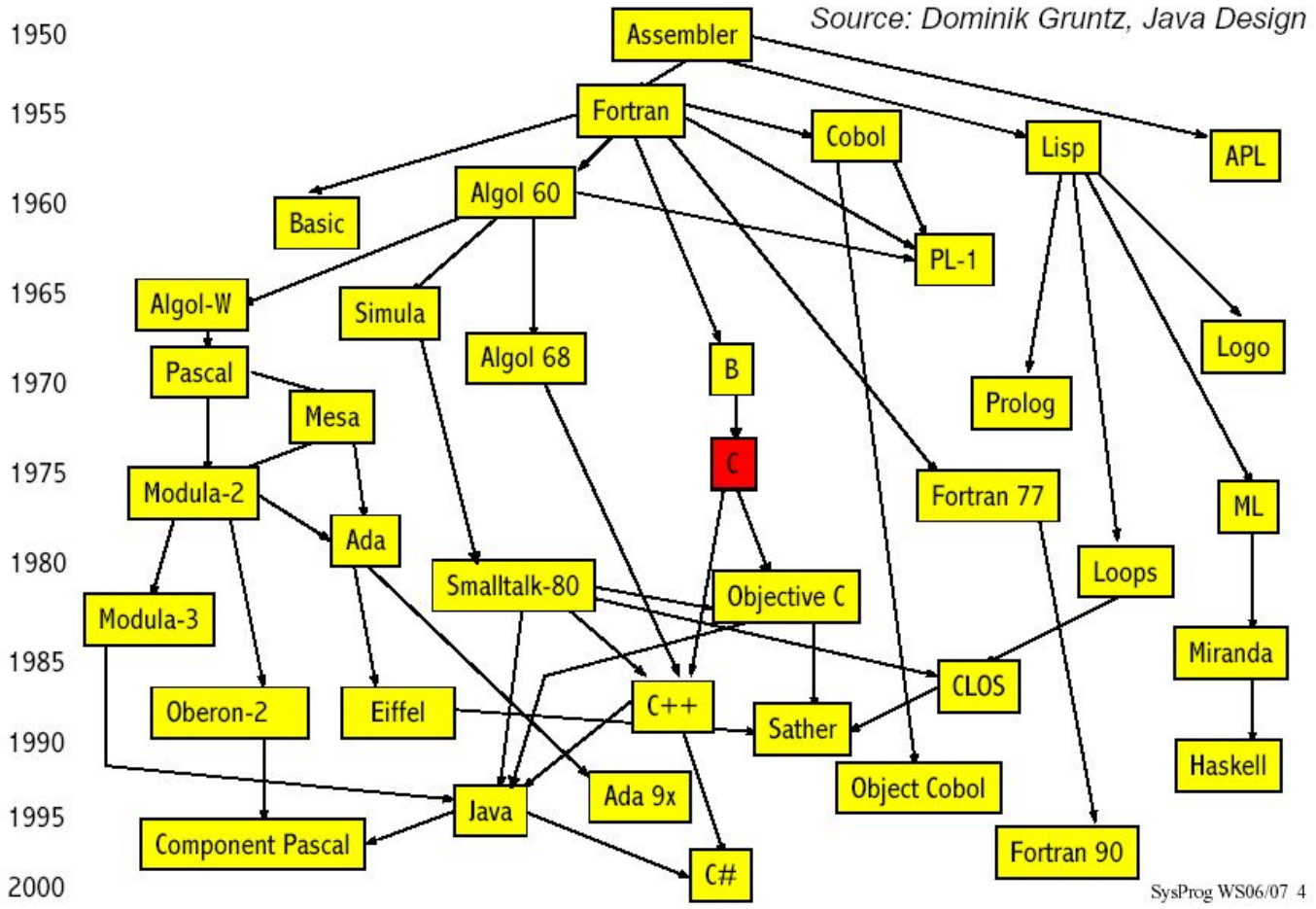
# Популярность ЯП

- RedMonk’s language ranking for February 2012



# История

- Развитие языков программирования



# История

- Восходит к языку **B** (К.Томпсон), который восходит к языку **BCPL** (Д. Мартин, 1967)
- Разработан между 1969 и 1973 годами вместе с ОС Unix
- Основной вклад в разработку принадлежит **Деннису Ритчи** (Dennis Ritchie)
- Изначально предназначался для системного программирования, т.е. создания
  - Операционных систем
  - Компиляторов
  - Утилит (служебных программ)

Деннис Ритчи

# История

- Разработка велась на микроЭВМ DEC PDP-11
  - Оперативная память – 24Кб
  - Из них используется ОС – 12 Кб
- Задачи
  - Разработка нового языка программирования
  - Разработка на нем операционной системы

# История

Д. Ритчи (слева) и К.Томпсон (справа) перед PDP-11 с двумя текстовыми терминалами (1972)

Фото с домашней страницы Д. Ритчи: *http://www.cs.bell-labs.dom/who/dmr/*

# История

- Первый стандарт (описание языка) опубликован Д.Ритчи и Б. Керниганом в 1978 (**K&R-C**)
- В слегка модифицированном виде язык был стандартизован ANSI в 1989 г. (**ANSI-C**, **C89**, **C90**)
- Международный стандарт языка утвержден ISO в 1999 г. (**C99**)
  - См. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf
  - Новые возможности
    - встраиваемые функции (объявленные с ключевым словом inline)
    - Сняты ограничения на место описания переменных
    - Несколько новых типов данных, включая long long int, bool, complex)
    - Массивы переменной длины
    - Поддержка однострочных комментариев, начинающихся с //, как в BCPL или C++
    - Новые библиотечные функции

# История

- Наиболее современный стандарт –
  **C11** или **ISO/IEC 9899:2011** (неофициально **C1X**)
  - См. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
  - Новые возможности
    - Выравнивание данных
    - Спецификатор функции _Noreturn;
    - Выражения, не зависящие от типа (Type-generic expressions) с использованием ключевого слова _Generic.
    - Поддержка многопоточности, добавленспецификатор типа _Thread_local, заголовочный файл <threads.h>, квалификатор типа _Atomic и заголовочный файл <stdatomic.h>
    - Функция getsФункция gets, признана устаревшей и заменена безопасной альтернативой gets_s;
    - Интерфейсы для проверки границ массива
    - …

# Простейшая программа

Функция ***main*** – без значения

главная (основная) функция всегда имеет имя ***main***

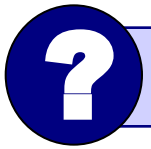
```
void main()
{

}
```

начало программы

«тело» программы (основная часть)

конец программы



**Что делает эта программа?**

# Вывод текста на экран

***include*** = включить

файл ***stdio.h***: описание стандартных функций ввода и вывода

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

вызов стандартной функции ***printf = print format*** (форматный вывод)

этот текст будет на экране

# Ждем нажатия любой клавиши

```
#include <stdio.h>

void main()
{
  printf("Привет!"); // вывод на экран
  getchar(
  );   /* ждать нажатия клавиши */
}
```

комментарий до конца строки

ждать ввода символа

комментарий между /* и */

# Переход на новую строку

```
#include <stdio.h>

void main()
{
  printf("Привет,\nВася!");
  getchar();
}
```

последовательность
\n (код 10)
переход на новую строку

на экране:

```
Привет,
Вася!
```

# Домашнее задание

1. Самостоятельно вспомнить/усвоить основные понятия алгоритмизации

2. Зарегистрироваться на **portal.edu.asu.ru**
(на первом практическом занятии)

3. Посмотреть ролик
“Знакомство с Microsoft Visual Studio 2008”

4. В книге
Дейтел Х. М., Дейтел П. Дж. Как программировать на C
прочитать Главы 1-3.

5. Установить дома компилятор и среду программирования и выполнить упражнения

# Вопросы?

- Быстрое введение в язык Си
  - Почему Си?
  - История
  - Простейшая программа
  - Переменные и типы
  - Операции и функции
  - Ввод/вывод
- Основы алгоритмизации
  - Алгоритм: определение
  - Алгоритм: свойства
  - Алгоритм: исполнитель
  - Алгоритм: формы записи
  - Алгоритмические структуры
  - Программы: виды ошибок
  - Языки программирования


Н.Копейкин *Битва снеговиков с углевиками*