Министерство образования и науки Российской Федерации
САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Н. Б. Культин

ТЕОРИЯ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Учебное пособие

Санкт-Петербург
2022

УДК 681.1
ББК 32.973
К90

Культин Н.Б. **Теория и технологии программирования**: Учеб. пособие. СПб., 2022 – 111 с., ил.

Учебное пособие представляет собой опорный конспект лекций по дисциплине «Теория и технологии программирования». Излагаются основы теории и практики программирования на языках высокого уровня. Рассмотрены этапы разработки компьютерной программы, алгоритмические структуры, базовые структуры данных и работа с ними, основы объектно-ориентированного программирования. Уделено внимание технологиям визуального проектирования и событийного программирования, разработке Windows Forms приложений на языке программирования C#.

Пособие предназначено для студентов, изучающих основы программирования в рамках дисциплины «Теория и технологии программирования».

Оглавление

Программное обеспечение

- Системное
- Прикладное (application)
- Инструментальное

Этапы разработки программы

- Постановка задачи (анализ требований, спецификация) – 20%
- Проектирование – 15%
- Кодирование – 20%
- Отладка – 20%
- Тестирование – 25%

Анализ требований и спецификация (пример)
Задача: разработать приложение, вычисляющее массу полого стержня, который может быть из разных материалов.
Как задать размер стержня и в каких единицах?
(R1, R2, L или D1, D2,L или R, d, L или D, d, L)
Как задать материал?
В каком виде вывести результат расчета?

Программа

- Исходная
- Выполняемая

Языки программирования

- Низкого уровня (машинно-ориентированые): машинный код, мнемокод, ассемблер (PDP-11, Intel x86, ARM, ...)
- Высокого уровня: Pascal, C/C++, C#, Python, ...

Ассемблер

PDP-11

DEC – Digital Equipment Corporation, 1970-80х, мини- и микро-ЭВМ

```
      .TITLE HELLO WORLD
      .MCALL .TTYOUT,.EXIT
HELLO:: MOV #MSG,R1        ; R1 - УКАЗАТЕЛЬ НА ТЕКУЩИЙ СИМВОЛ
   1$: MOVB (R1)+,R0       ; СИМВОЛ В R0Л
      BEQ DONE             ; ЕСЛИ СИМВОЛ НОЛЬ, ВЫХОД
      .TTYOUT              ; ПЕЧАТЬ СИМВОЛА
      BR 1$                ; СЛЕДУЮЩИЙ СИМВОЛ
  DONE:.EXIT
  MSG: .ASCIZ /Hello, world!/
     .END HELLO
```

PDP-11

```
; обработка сигналов с датчиков объекта управления
; запуск каждые 4 мс
BUF11: .WORD 0
BUF12: .WORD 0
BUF13: .WORD 0
P1: .WORD BUF11

ST:MOV P1,R1
  MOV @#160120,(R1)+
  CMP R1, BUF13
  BLE 1$
  MOV #BUF11,R1
1$:MOV #BUF1, P1
```

x86

Intel 8086 – 1978

```
HELLO DB 'Hello, World!'
LEA DX, HELLO
MOV AH, 9
INT 21H

GET_KEY:
   MOV AH,1
   INT 21H
   CMP AL, 'Y'
   JE YES
   CMP AL, 'N'
   JE NO
   JNE GET_KEY

YES:  .....

NO:  ....
```

Ассемблер

Где?

- Элементы ОС
- Системы реального времени (управляющиепрограммы)

+Эффективный код (малый объем, высокая скорость работы)

- Необходимость знания архитектуры процессора, большое количество команд, низкая скорость разработки

Языки программирования высокого уровня

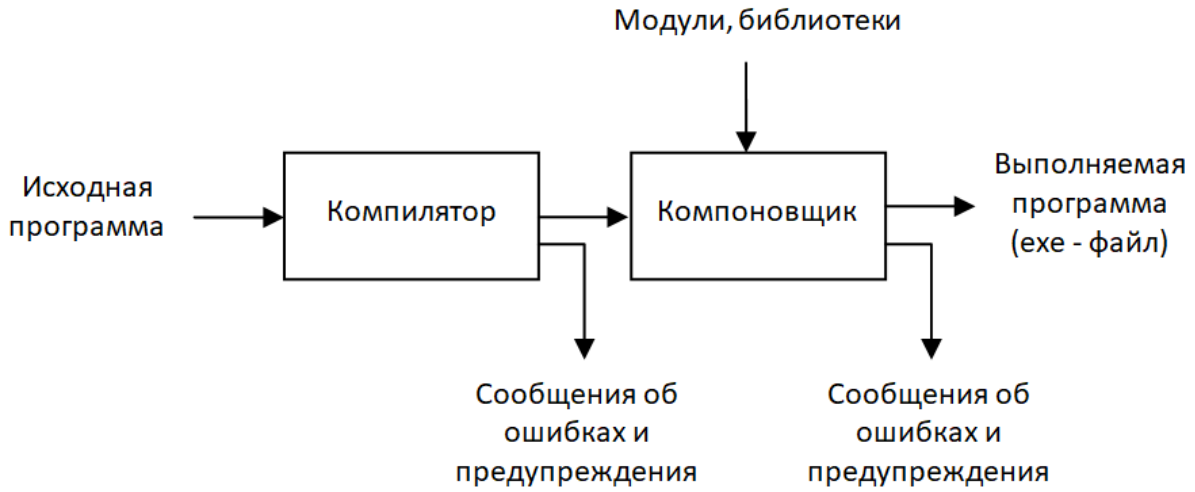
- Универсальные: BASIC, Pascal /Delphi, C/C++, C#, Obective-C, Java, Pyhton, ...)
- Специальные (LISP, Prolog, VBA, R, JScript, VBScript, PHP, ...)

- Компилируемые: Pascal, C, C++, Obective-C, ...
- Интерпретируемые: (BASIC, Python, VBA, R, JScript, VBScript, PHP,, ...)

FORTRAN	1958	FORmula TRANslator. Инженерные расчеты, IBM
Algol	1958	ALOritmic Language. Algol-68
COBOL	1960	Common Busines Oriented Language
BASIC	1963	BASIC – Beginner's All-purpose Symbolic Instruction Code (универсальный код символических инструкций для начинающих). Джон Кеммени и Томас Курц. В 70-х годах язык стал основным языком для микрокомпьютеров.
Pascal	1968	Н. Вирт. Язык создавался как средство изучения теории и практики программирования. При разработке языка были учтены требования структурного программирования.
C	1972	Деннис Ричи. Создавался как альтернатива ассемблеру (в 1968 году Ричи работал в «Белл телефон лабораториес» и создавая свой язык, надеялся, что он будет полезен при программировании новой операционной системы «Юникс»)
Turbo-Pascal	1983	Филипп Кан разработал быстрый и дешевый (49.99$) компилятор языка Паскаль. В первые два года было продано более 300 тысяч копий, что вывело фирму «Борланд» в разряд основных производителей программного обеспечения.
C++	1983	Бьярн Страуструп. Язык создавался и совершенствовался на протяжении нескольких лет как расширение языка С в сторону объектно-ориентированного программирования.
Prolog	1972	Алэнн Кольмерон. Пролог – язык логического программирования. Используется для программирования задач принятия решений, экспертных систем.
Visual Basic		Язык разработки Win32 приложений для Windows
VBA - Visual Basic for Application		Язык разработки «офисных» (встроенных) приложений (надстроек, макросов), расширяющих функциональные возможности офисных приложений Microsoft (Word, Excel, PowerPoint, Outlook, Project).
Java		Универсальный, платформонезависимый язык программирования. Широко используется в WEB программировании. 1995, Sun Microsystems, Джеймс Гослинг.
Delphi		Основан на языке Object Pascal. Используется в среде Delphi для разработки Win32 приложений.
VBScript, JScript,		Интерпретируемые языки программирования основанные, соответственно на BASIC и Java. Используются в WEB программировании (технология ASP)

VB .NET		Основан на языке Visual Basic. Используется в среде Microsoft Visual Basic .NET для разработки .NET приложений
C#		Основан на языке Java. Используется в среде Microsoft Visual C# для разработки .NET приложений. разработан в компании Microsoft в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга
Python		Интерпретируемый язык, минимальный синтаксис, большое количество библиотек различного назначения
R		Интерпретируемый язык и среда разработки, ориентированные на статистическую обработку и представление данных.

Компиляция



Ошибки

- Времени компиляции (синтаксические)
- Времени выполнения или исключения (неверные исходные данные, ошибки алгоритма, ...)

Среда разработки

- Редактор кода
- Компилятор
- Отладчик
- ...

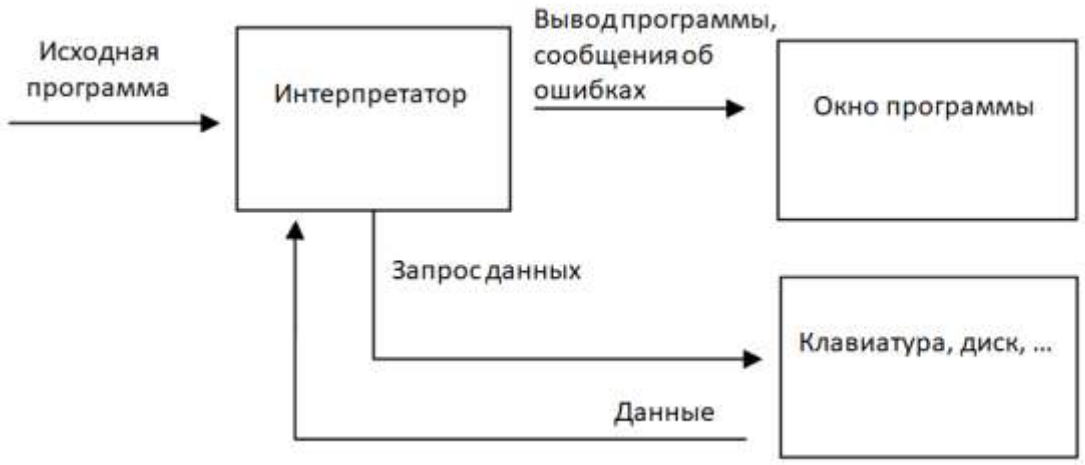
Среда разработки

- Microsoft Vicual Studio (C++, C#, Visual Basic)
- QT Creator (C, C++)
- Android Studio (Java)
- Embarcadero RAD Studio (Delphi, C++)
- Xcode (Objective-C)

Интерпретация

Среда разработки-выполнения

- Редактор кода
- Интерпретатор

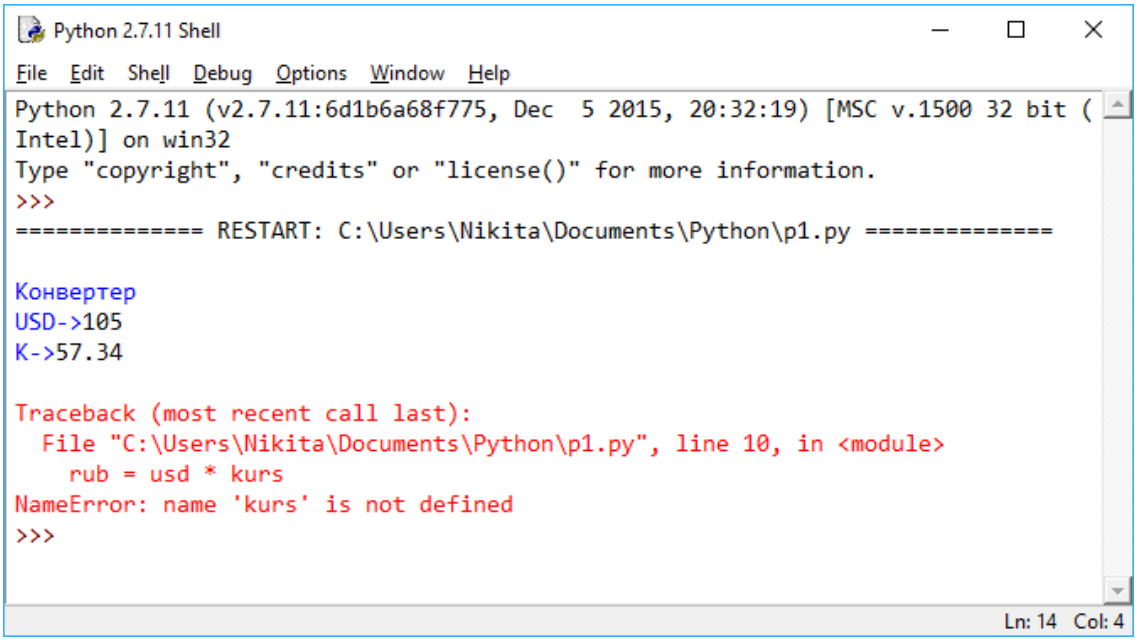


p1.py - C:\Users\Nikita\Documents\Python\p1.py (2.7.11)

File Edit Format Run Options Window Help

```
print '\nКонвертер'

st=raw_input('USD->')
usd = float(st)

st=raw_input('K->')
k = float(st)

rub = usd * kurs
print usd,'$ = ', rub, 'руб.'
```

Ln: 10 Col: 16



Компиляция и интерпретация

	Компиляция	Интерпретация
Синтаксические ошибки	Обнаруживаются в процессе компиляции	Обнаруживаются во время выполнения (работы) программы
Выполняемая программа (.exe файл)	Создается	Не создается
Возможность запуска программы	Из среды разработки (отладка) или из операционной системы	Только из среды разработки
Требуется среда разработки для запуска программы на компьютере пользователя	НЕТ	ДА
Скорость работы программы	Высокая	Низкая
Потребность в ресурсах	Высокая	Низкая

Алгоритм

Алгоритм – описание действий, которые надо выполнить для достижения поставленной задачи

- Массовость
- Однозначность
- Результативность

Алгоритм

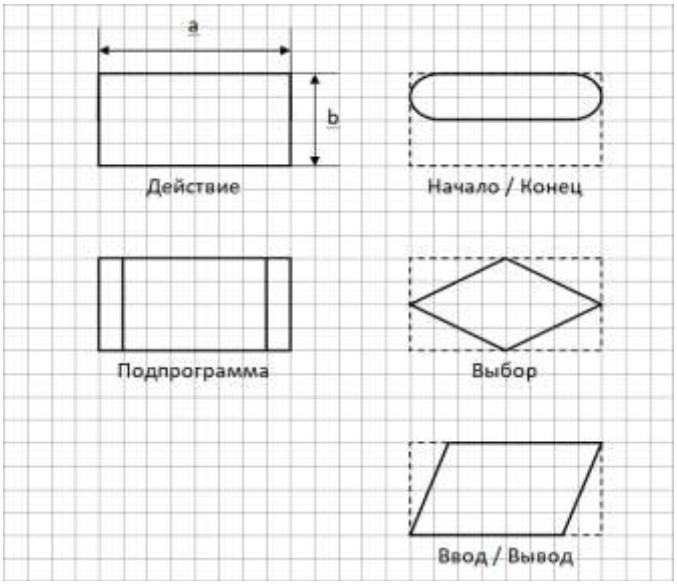
- Словесное описание
- Блок-схема
- Язык программирования
- Язык проектирования задач

Словесное описание

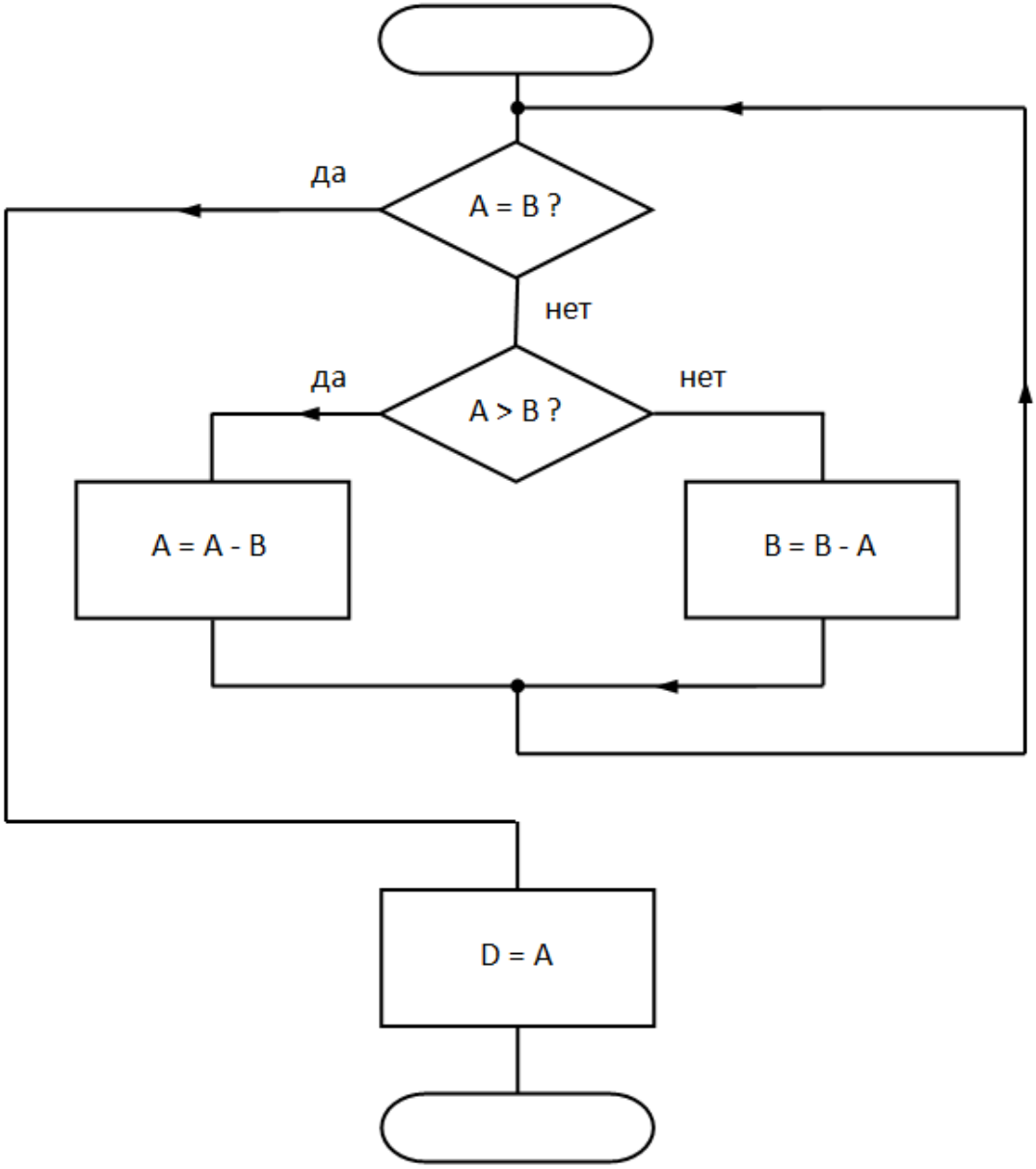
Алгоритм Евклида
(вариант 1)
Если числа равны, то наибольший общий делитель равен любому из этих чисел. Если числа не равны, то надо из одного числа вычесть другое и присвоить полученное значение этому числу. Продолжить сравнивать и вычитать до тех пор, пока числа не станут равными.
Алгоритм Евклида
(вариант 2)
Если числа равны, то наибольший общий делитель - первое число. Если числа не равны, то надо из большего числа вычесть меньшее число и заменить большее число полученным значением. Если полученные таким образом новые значения не равны между собой, продолжить сравнивать и вычитать числа до тех пор, пока они не станут равными.

Блок-схема

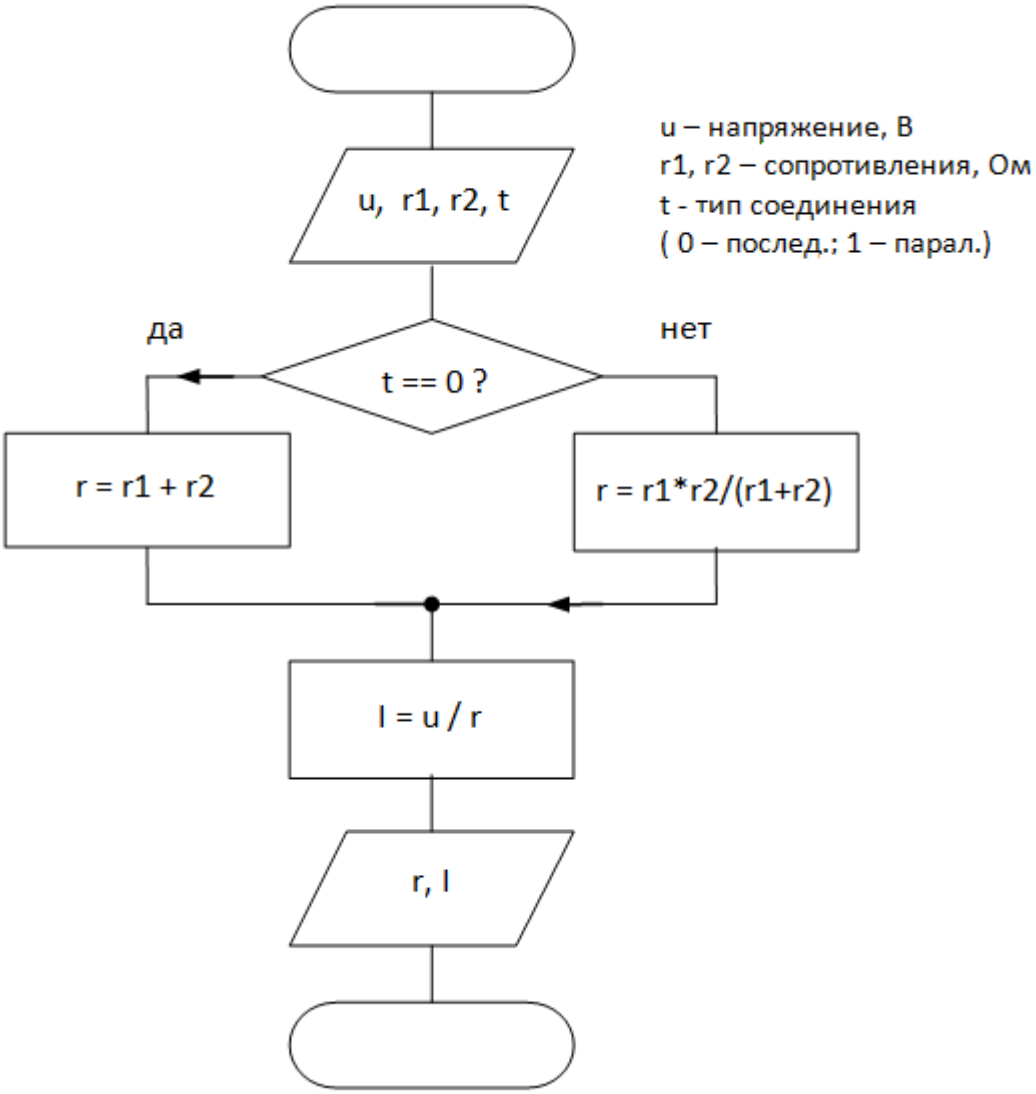
- Начало / конец
- Ввод / вывод
- Действие (обработка)
- Выбор (решение)
- Подпрограмма (функция)

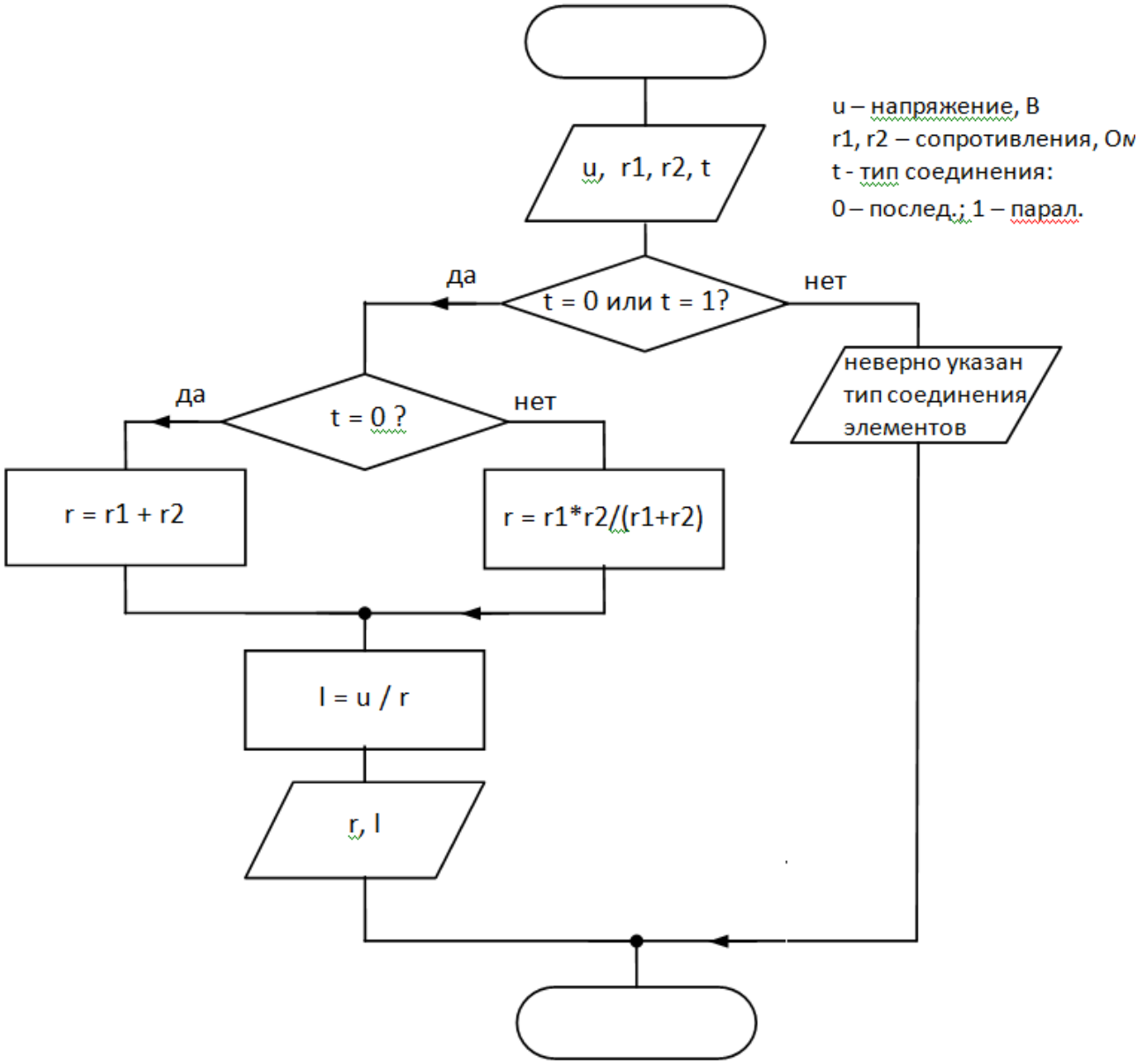


Алгоритм Евклида



Неоднозначность





Правила изображения алгоритмов

- Все однотипные элементы должны быть одинакового размера!!!
- Стрелки – справа налево, снизу вверх
- Начало, Конец – НЕТ
- Да, Нет или true, false - !!!
- Точки объединения!!!
- Вход/выход в элемент Действие сбоку – допускается.

Язык программирования

```
// НОД ПОЛОЖИТЕЛЬНЫХ чисел
while (a != b)
{
        if (a > b)
        a = a - b;
```

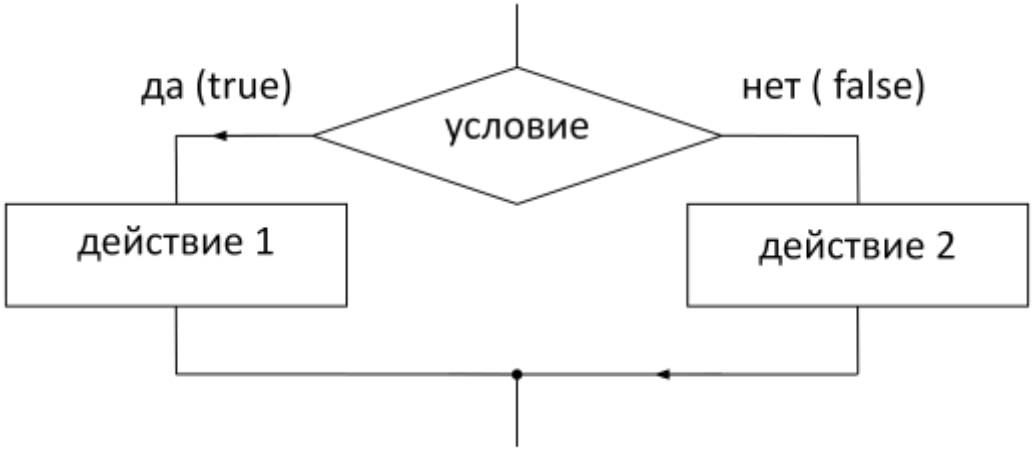
```
  else
        b = b - a;
}
d = a;
```

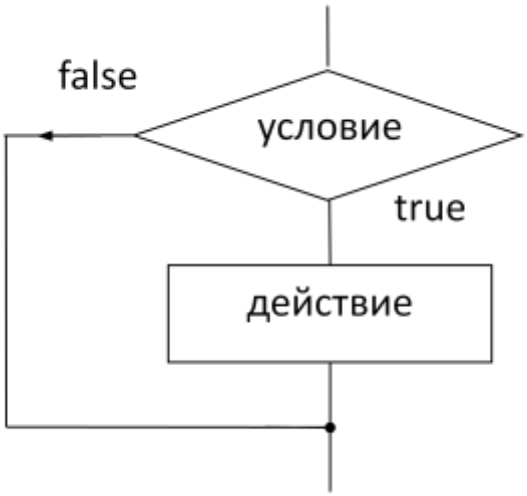
Алгоритмические структуры

- Следование
- Выбор варианта
- Цикл

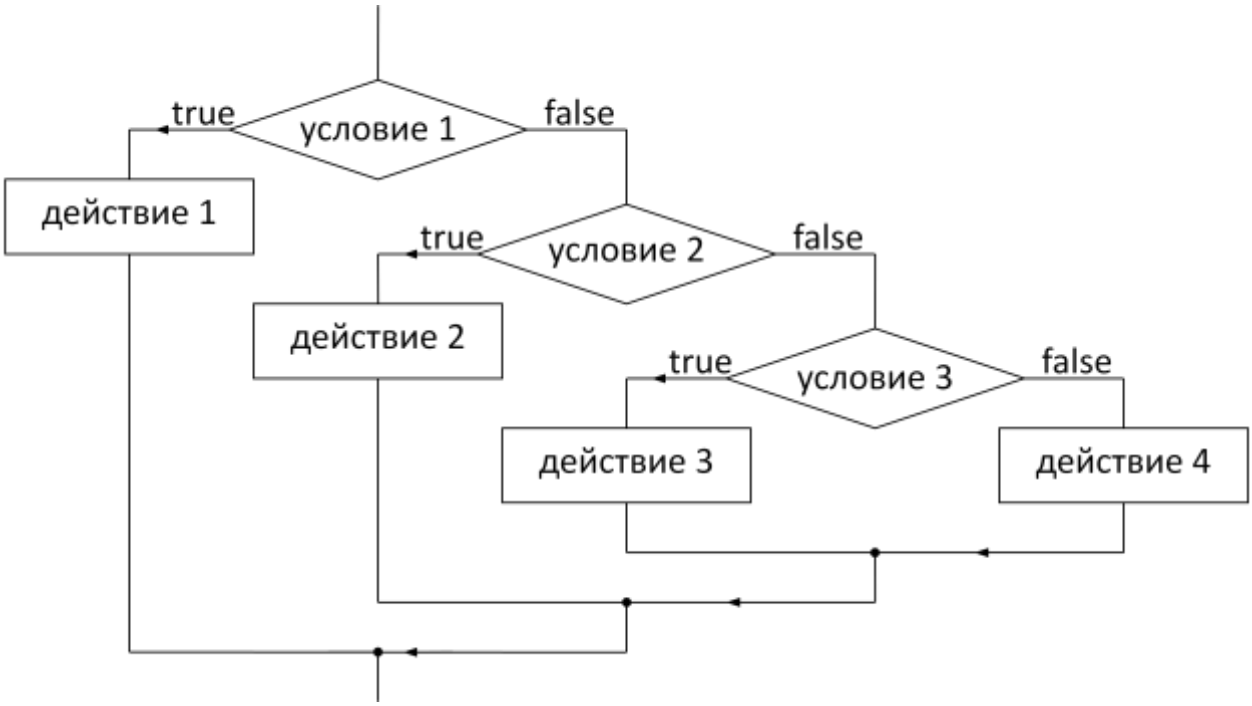
Следование

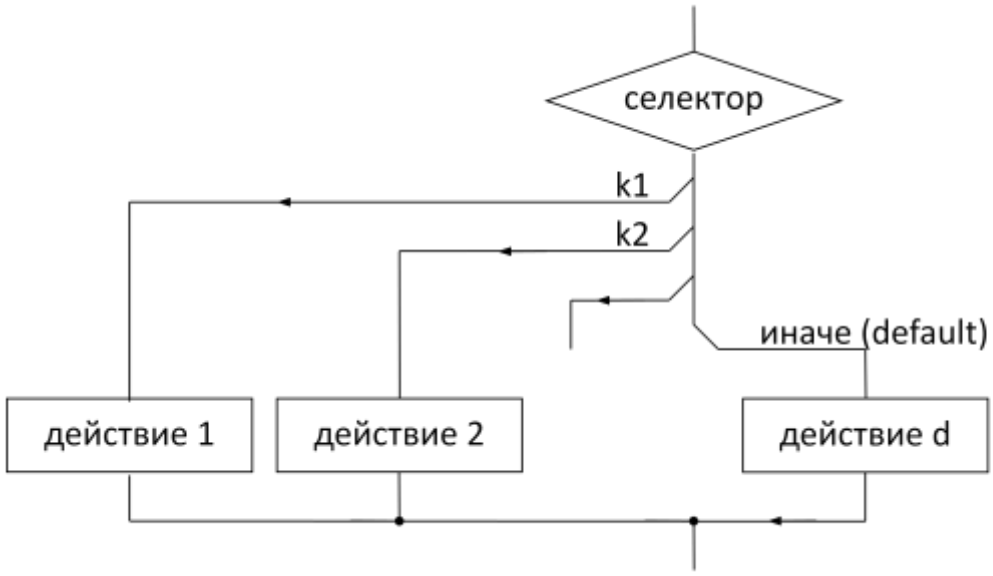
Выбор



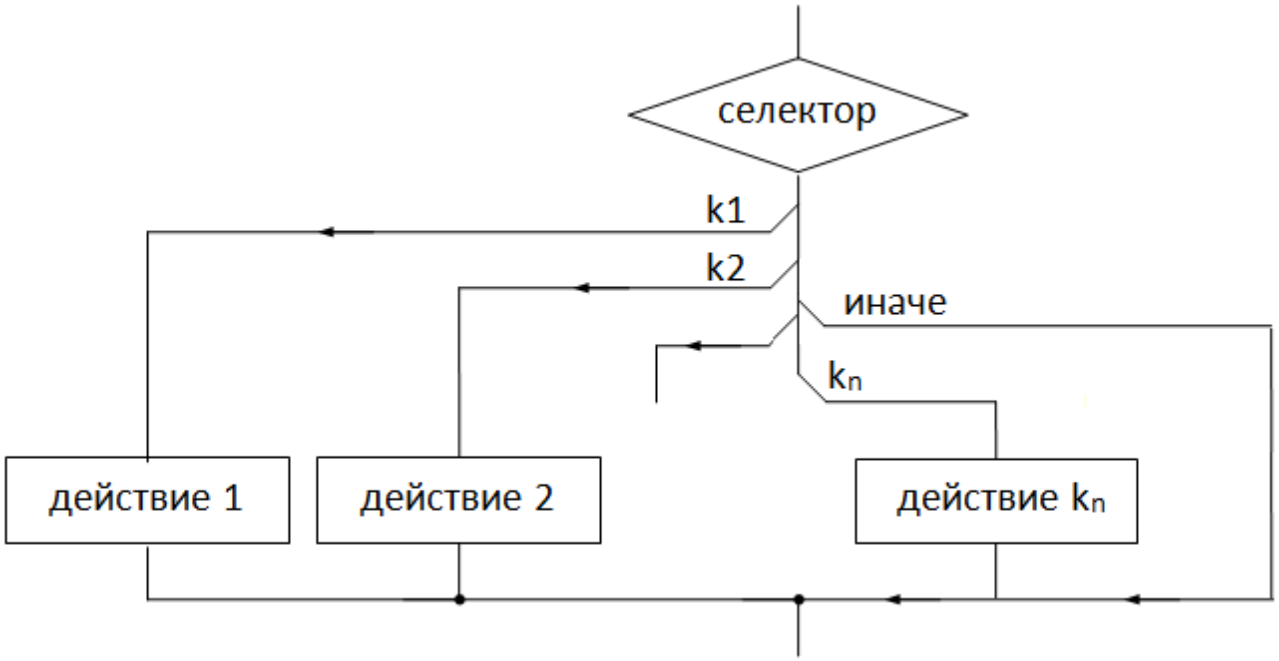


Множественный выбор





switch (*селектор*)
{
case *v1*: *инструкции*; **break**;
case *v2*: *инструкции*; **break**;
case *vk*: *инструкции*; **break**;
default: *инструкции*; **break**;
}

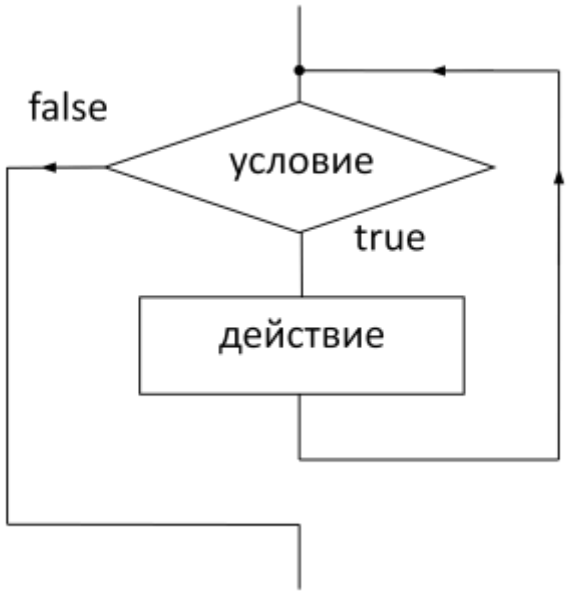


switch (*селектор*)
{
case *v1*: *инструкции*; **break**;
case *v2*: *инструкции*; **break**;
case *vk*: *инструкции*; **break**;
}

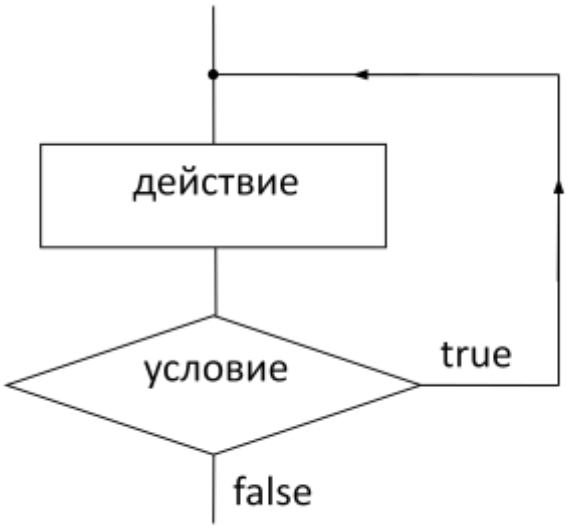
Циклы

- С предусловием (while)
- С постусловием (do while)
- Фиксированное количество повторений (for)

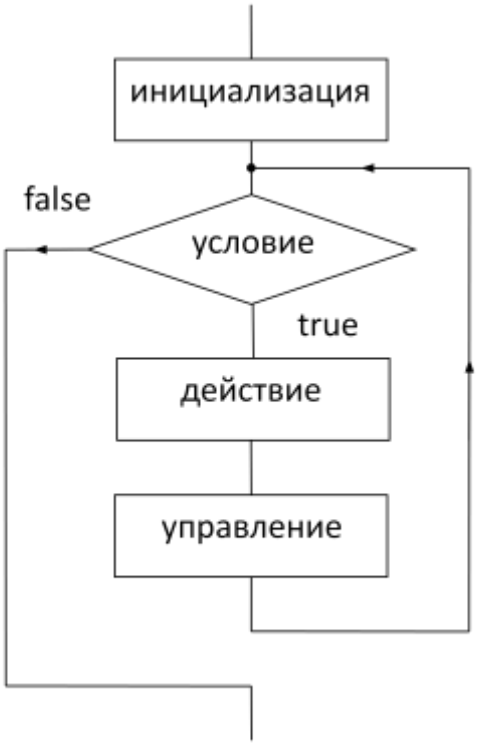
while



do while



for



Условие

Условие – выражение логического типа.

Условие

- простое
- сложное

Простое условие

оп1 ос оп2

оп – операнд
ос – оператор сравнения
ос: >, <, ==, !=, >=, <=

Сложное условие

оп1 ло оп2
оп – операнд, выражение логического типа
ло – логический оператор
ло: &&, ||

! оп

Примеры
(x >= x1) && (x<=x2)
!((x < x1) || (x > x2))

D >= 0
(D > 0) || (D == 0)
! (D < 0)

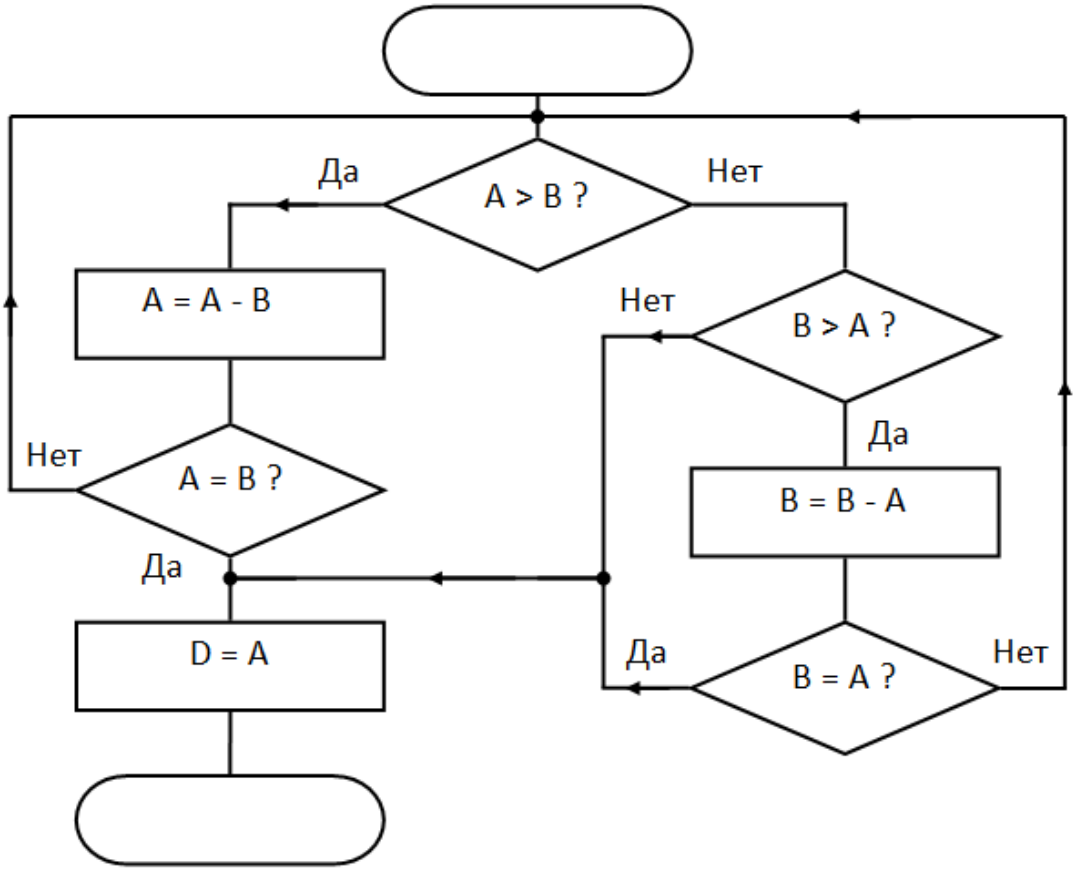
(ch >= '0') && (ch <= '9')
!((ch >= '0') && (ch <= '9'))
(b >= 'A') && (b <= 'я')

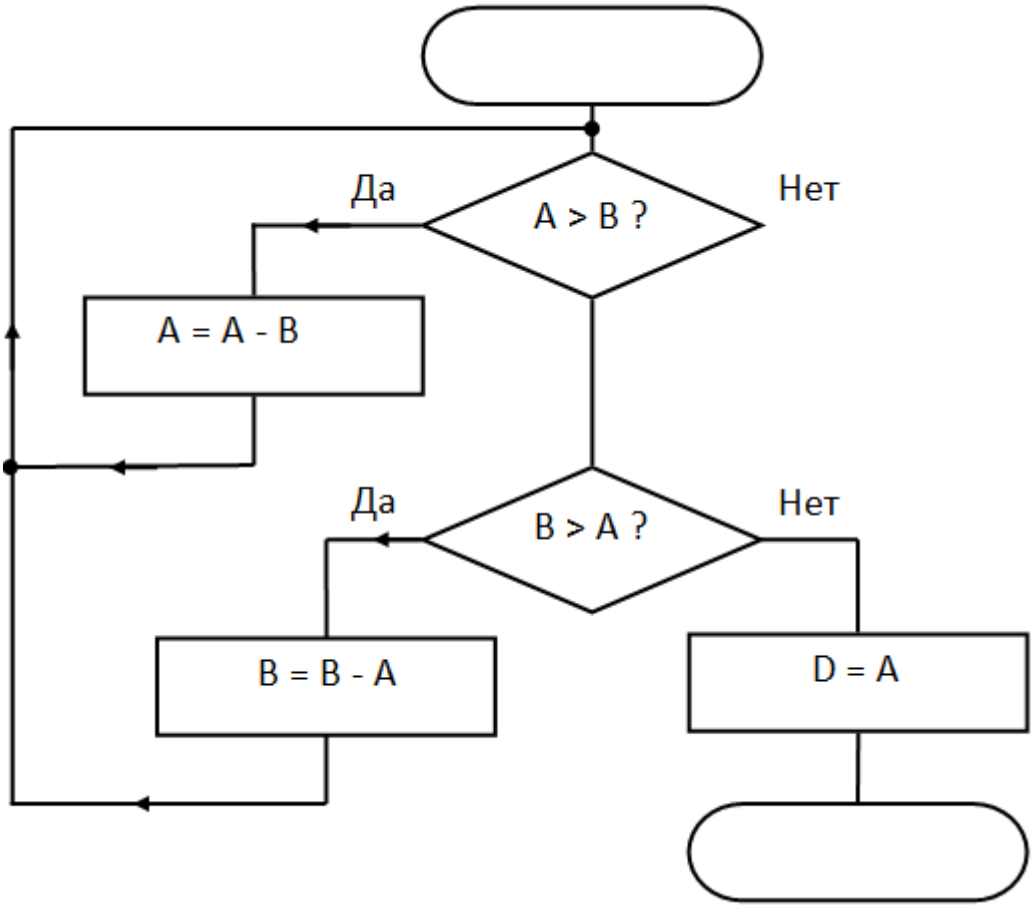
Структурное программирование

Дал (O.-J. Dahl), Дейкстра (E. W. Dijkstra), Хоор (C.A.R. Hoare)
Метод программирования, обеспечивающий создание программ, структура которых ясна и неразрывно связана со структурой решаемой задачи.

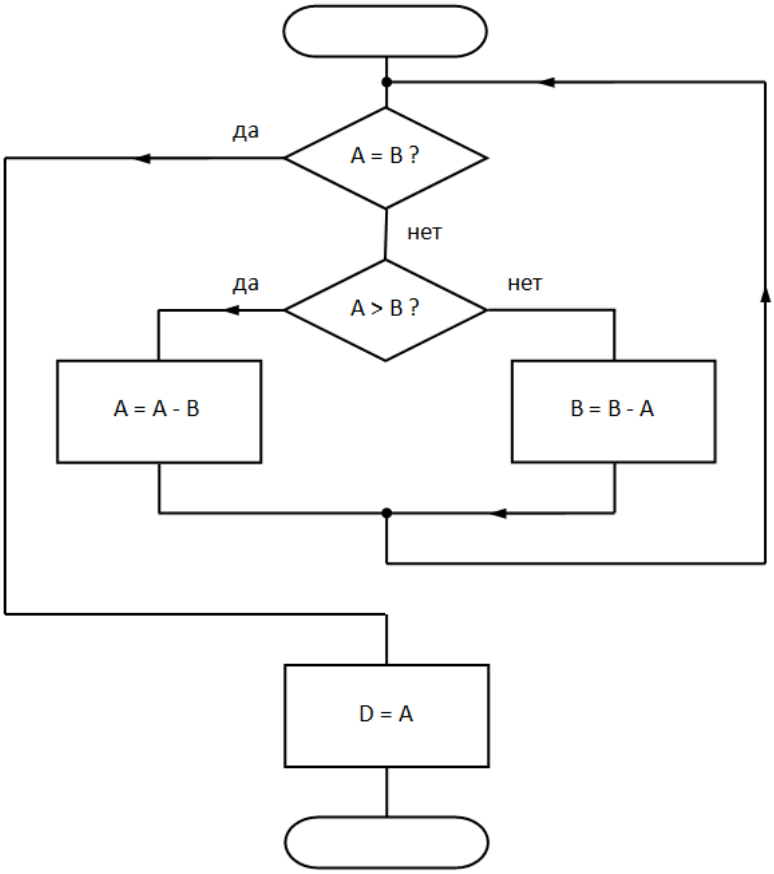
- Структурное программирование
- Никаких трюков и заумного программирования
- Меньше переходов (goto, break, exit, halt)
- Выбор с использованием if-then-else
- Простота циклов (переход можно представить как цикл)
- Сегментация (разбиение задач на подзадачи)
- Рекурсия – разумное использование
- Содержательные обозначения

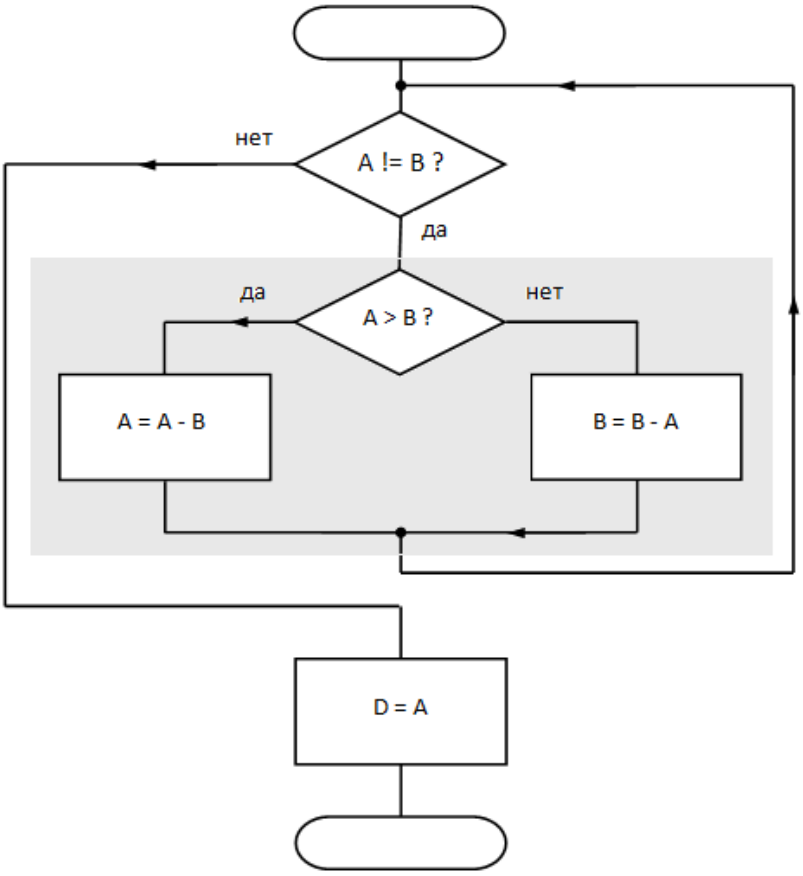
Bad!


A > B ?
Да
Нет
A = A - B
B > A ?
Нет
Да
A = B ?
Нет
B = B - A
Да
Да
D = A
B = A ?
Да
Нет


A > B ?
Да
Нет
A = A - B
B > A ?
Да
Нет
B = B - A
D = A

Cool!





Стиль программирования

Стиль - совокупность признаков и особенностей.

Хороший стиль программирования:

- Имеющие смысловую нагрузку имена переменных, функций и др. объектов
- Комментарии
- Отступы
- Пустые строки

Пример-bad

```
double p;
Console.Write("Сумма, руб. >> ");
double sum = System.Convert.ToDouble(Console.ReadLine());
Console.Write("Срок, мес. >> ");
int s = System.Convert.ToInt32(Console.ReadLine());
if (sum < 10000) p = 8.0;
else if (sum < 50000) p = 8.5;
else if (sum < 100000) p = 9.0;
else p = 10.5;
double d = sum * (p / 12 / 100) * s;
double total = sum + d;
```

Пример-cool

```
double sum;    // сумма вклада
      int period;    // срок вклада, мес.

      double pr;     // процентная ставка
      double profit; // доход
      double total;  // сумма в конце срока

      Console.Write("Доход\n");

      Console.Write("Сумма, руб. >> ");
      sum = System.Convert.ToDouble(Console.ReadLine(););

      Console.Write("Срок, мес. >> ");
      period = System.Convert.ToInt32(Console.ReadLine());

      if (sum < 10000)
         pr = 8.0;
      else
         if (sum < 50000)
            pr = 8.5;
         else
            if (sum < 100000)
               pr = 9.0;
```

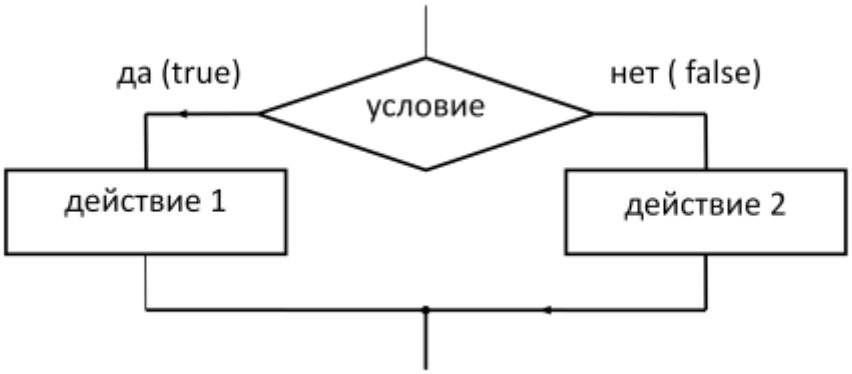
```
        else
            pr = 10.5;

    // т.к. ставка процентов годовых,
    // а срок вклада в месяцах, вычислим процентов в месяц
    profit = sum * (pr / 12 / 100) * period;
    total = sum + profit;
```

Язык проектирования задач/псевдокод

Алгоритмические структуры

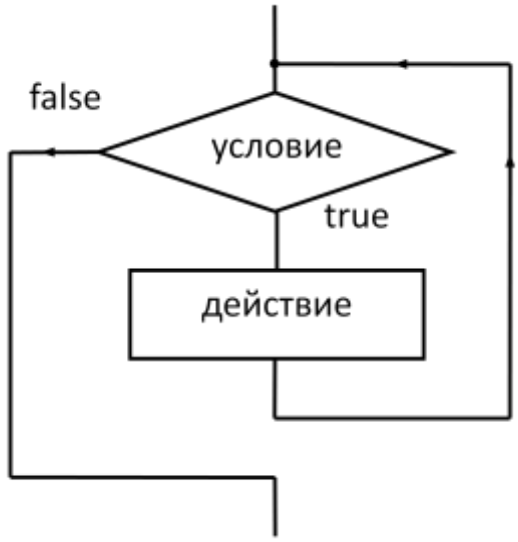
- Следование
- Выбор
- Цикл

Выбор



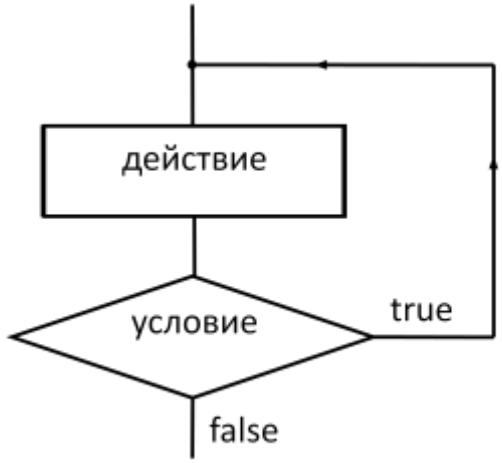

```
if условие then
        действие1
else
        действие2
```

Цикл с предусловием




```
while условие do
        действие
end
```

Цикл с постусловием




```
do
  действие
while условие
```

Цикл for


```
for k=1 to n do
        действие
end
```

Ввод, вывод

```
read(список_переменных)

write(список_переменных)
write(сообщение)
```

Данные

```
Тип данных:
        целое, веществ, строка, символ
или
        int, float, string, char

Переменная:  тип имя
Массив:  тип имя[размер]

float diam, wall, len; // диаметр, толщина стенки, длина
int n; // код материала
string mat;
float pl, v, m;  // плотность материала, объем и масса стержня

write (1 – Сталь, 2 – Медь, 3 – Алюминий)
read (n) // код материала, введенный пользователем
if  n <1 или n > 3 then
   write(Неверно указан номер материала)
else
   read (diam, wall, len); // размеры стержня
```

```
if n = 1 then mat = Сталь; pl = 7856
  else
    if n = 2 then
      mat = Медь; pl = 8990
    else
      mat = Алюминий; pl = 2712
v = 3.14*(diam*wall -  wall)*vall*len
m = pl * v
 write(diam, wall, len, v, m)
```

```c
// расчет массы полого стержня
int main()
{
    float diam, wall, len; // диаметр, толщина стенки, длина
    int n;                 // выбор пользователя (код материала)
    char mat[12];
    float pl, v, m;  // плотность материала, объем и масса стержня

    printf("1 – Сталь\n2 – Медь\n3 – Алюминий\n");
    printf(">>");
    scanf("%i",&n); // номер материала, введенный пользователем

    if (n < 1 || n > 3) {
        printf("Неверно указан номер материала");
    }
    else
    {
        printf("Введите в мм диаметр, толщину стенки и длину стержня");
        scanf("%f%f%f", &diam, &wall, &len); // размеры стержня
```

```
        if (n == 1) {
            strcpy(mat, "Сталь"); pl = 7856;
        }
        else {
            if (n == 2) {
                strcpy(mat, "Медь"); pl = 8990;
            }
            else {
                strcpy(mat, "Алюминий"); pl = 2712;
            }
        }

        v = 3.14*wall*(diam - wall)/1000000000;
        m = pl * v;

        printf("Диаметр: %6.3f мм Толщ.стенки: %6.3f мм", diam, wall);
        printf("Длина: %6.3f Материал: %s Масса: %6.3f кг",
                                                    len, mat,  m);
    }

    puts("Press any key ...");
    int ch = getchar(); ch = getchar();
    return 0;
}
```

```
static void Main(string[] args)
{
    double diam, wall, len; // диаметр, толщина стенки, длина
    int n;                  // выбор пользователя (код материала)
    string mat;
    double pl, v, m;  // плотность материала, объем, масса

    Console.Write("1 - Сталь\n2 - Медь\n3 - Алюминий\n");
    Console.Write(">>");
    n = Convert.ToInt32(Console.ReadLine()); // номер материала

    if ((n < 1) || (n > 3))
        Console.Write("Неверно указан номер материала");
    else
    {
        Console.Write("Диаметр >");
        diam = Convert.ToDouble(Console.ReadLine());

        Console.Write("Толщина стенки >");
        wall = Convert.ToDouble(Console.ReadLine());

        Console.Write("Длина >");
        len = Convert.ToDouble(Console.ReadLine());
```

```
        if (n == 1) {
              mat = "Сталь"; pl = 7856;
         }
         else
           if (n == 2) {
              mat = "Медь"; pl = 8990;
           }
           else {
               mat = "Алюминий"; pl = 2712;
           }

         v = 3.14 * wall * (diam - wall) * len / 1000000000;
         m = pl * v;

         Console.Write("Диаметр: {0:f3} мм Толщ.стенки: {1:f3}",
                       diam, wall);
         Console.Write("Длина: {0:f3} мм Материал: {1} Масса: {2:f3} кг",
                        len, mat, m);
      }

      Console.Write("\nPress any key ...");
      Console.ReadKey();
 }
```

Тип данных

Стандартные простые
Целый

- Вещественный
- Символьный
- Строковый
- Логический

Целый

тип	диапазон	байт
byte	-128 … 127	1
Int32 (int)	-32768 … 32767	4
Int64	-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807	8

Вещественный

Вещественный:
Single (float) ±(3,402823x10-38 ... 3,402823x1038), 7-8 знач. цифр, 4 байта
Double±(1,79769313486232x10-308 ... 1,79769313486232x10308), 14-15 знач. цифр, 8 байт

Char

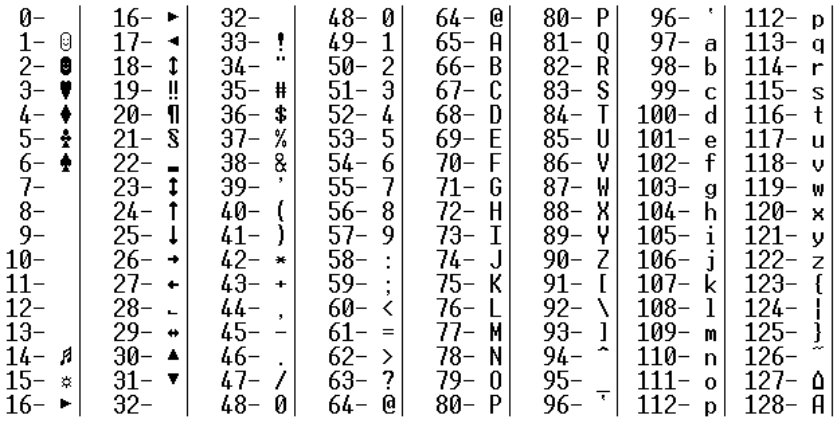
'0'<'1'<..'9' <'A'<'B' ..<'Z'<'a'<'b' ..<'z'<'A'<'Б' ..<'Я'

- ACCII
- ANSI
- UTF-8

ASCII

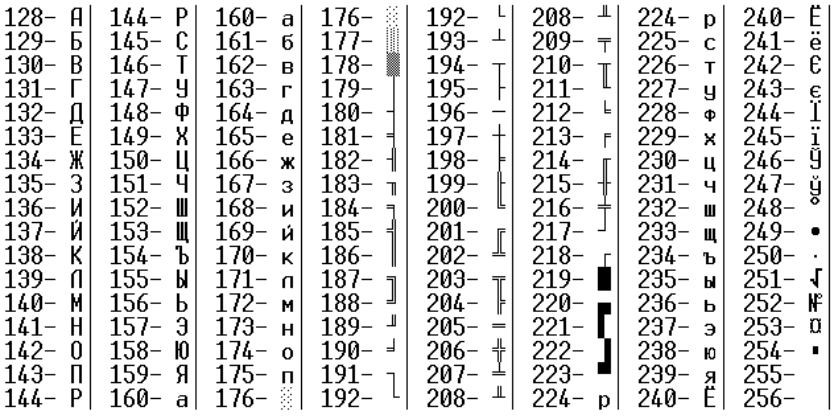
ASCII - American Standard Code for Information Interchange

0-127 – основной набор:



0-48, .. 9-57, … , A-65, .. Z-90, a-97, .. z-122

128 - 255 – нац. и доп. символы:



А-128, .. Я-159, а-160, .. п-175, р-224, .. я-239

Специальные символы

8 – Backspace

10 - CR

13 - LF

upcase

```
// преобразование строчных букв в прописные
char* upcase(char *st)
{
   int i = 0;
   while ( st[i] )
   {
      if (st[i] >= 'a' && st[i] <= 'z' ||   // латинские
```

```
            st[i] >= 'а' && st[i] <= 'п')  // русские
         st[i] -= 32;
      else if (st[i] >= 'р' && st[i] <= 'я')
            st[i] -= 80;
      i++;
   }
   return st;
}
```

ANSI (Win1251)

0-127 – основной набор:

0-48, .. 9-57, A-65, .. Z-90, a-97, b-98, .. z-122

128 - 255 – нац. и доп. символы:

А-192, .. Я-223, а-224, .. я-255

	ASCII	ANSI	ANSI - ASCII
А	128	192	64
Я	159	223	64
а	160	224	64
п	175	233	64
р	224	240	16
я	239	255	16

```
// преобразует ANSI строку в ASCII строку
char* rus(char st[])
{
   char* st2 = new char[strlen(st)  + 2];
   int i = 0;

   while (st[i] != 0)
   {
      unsigned char ch = st[i];
      if (( ch >= 192) && ( ch <= 255))
      {
          if ( ch < 240) // буква русского алфавита
                st2[i] = st[i] - 64;
          else
                st2[i] = st[i] - 16;
      }
```

```
      else  // остальные символы
              st2[i] = st[i];
      i++;
   }
   st2[i] = 0;
   return st2;
}

printf (rus("Привет, Барт!"));
```

UTF-8

UTF - Unicode Transformation Format
Символ – 1,2 или 3 байта

- основные символы (0-127) – 1 байт
- буквы русского алфавита – 2 байта

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	D082	D083	E2809A	D193	E2809E	E280A6	E280A0	E280A1	E282AC	E280B0	D089	E280B9	D08A	D08C	D08B	D08F
80	Ђ	Г	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
	D192	E28098	E28099	E2809C	E2809D	E280A2	E28093	E28094		E284A2	D199	E280BA	D19A	D19C	D19B	D19F
90	ђ	‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
	C2A0	D08E	D19E	D088	C2A4	D089	C2A6	C2A7	D081	C2A9	D084	C2AB	C2AC	C2AD	C2AE	D087
A0		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	-	®	Ї
	C2B0	C2B1	D086	D196	D291	C2B5	C2B6	C2B7	D191	E28496	D194	C2BB	D198	D085	D195	D197
B0	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	S	s	ї
	D090	D091	D092	D093	D094	D095	D096	D097	D098	D099	D09A	D09B	D09C	D09D	D09E	D09F
C0	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	D0A0	D0A1	D0A2	D0A3	D0A4	D0A5	D0A6	D0A7	D0A8	D0A9	D0AA	D0AB	D0AC	D0AD	D0AE	D0AF
D0	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	D0B0	D0B1	D0B2	D0B3	D0B4	D0B5	D0B6	D0B7	D0B8	D0B9	D0BA	D0BB	D0BC	D0BD	D0BE	D0BF
E0	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	D180	D181	D182	D183	D184	D185	D186	D187	D188	D189	D18A	D18B	D18C	D18D	D18E	D18F
F0	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Строковый

C#
string

C/C++
строка - массив символов

Логический

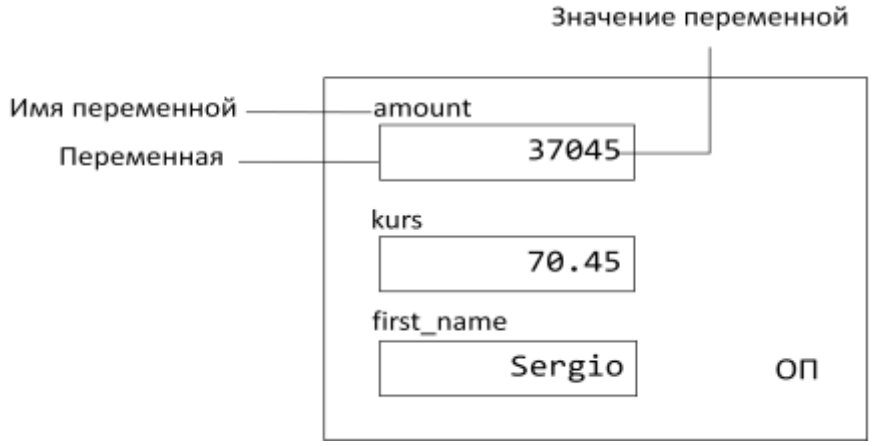
C#
bool

C/C++
bool

Переменная

ОП - оперативная память:

- сегмент кода, сегмент данных, сегмент стека
- "куча"



тип имя;
тип имя1, имя2, имя3, .. ;
тип имя = значение;

```
int t; // способ соединения элементов: 0 - посл., 1 - пар.

double u = 12.5; // напряжение, вольт
double r1, r2; // сопротивления, Ом

string Name = "Bart Simpson";
string First_Name;

char ch = 'y';

bool done = false;
```

Инструкция присваивания

имя = выражение;
Операнд1 Оператор Операнд2
или
Операнд
где: Операнд - переменная, константа, функция или выражение
Операторы: +, -, * , /

Выражение:

- тип
- значение

Оператор	Пример использования	Эквивалентная инструкция
++	i ++	i = i + 1
--	i --	i = i - 1
+=	n += 10	n = n + 10
*=	a*=b	a = a * b
/=	s/=n	s = s / n
%=	r %= 100	r = r % 100

Преобразование типов

```
int sum, n;
float mean;
int k;

mean = sum / n;   ?????

// C/C++
mean = (float) sum / n;
k = mean;

// C#
mean = Convert.ToSingle(sum) / n;
k = Convert.ToInt32(mean);
```

Ввод / вывод

- Консоль
- Файл
- Поток

Ввод данных

```
printf("Кол-во>");
scanf("%i",&n);

printf("Диаметр>");
scanf("%f", &diam);

Console.Write("Кол-во>");
st = ReadLine();
```

```
n = Convert.ToInt32(st);
Console.Write("Диаметр >");
st = ReadLine();
diam = Convert.ToDouble(st);
```

Вывод

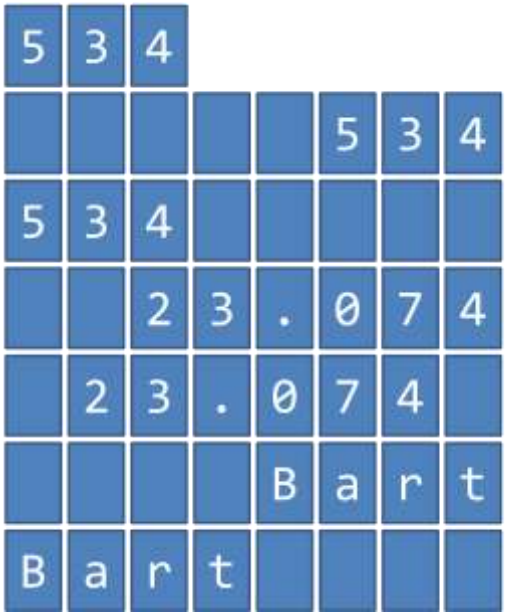
функция_вывода(строка_форматирования, список_выражений);

```
printf("Материал: %s\nМасса: %6.2f кг\n", mat,  m);
```

```
Console.Write("Материал: {0}\nМасса: {1:f2} кг\n", mat, m);
```

Некоторые форматы

	C/C++		**C#**	
	формат	пример	формат	пример
Целое	%i	%i	{n}	{0}
	%wi	%10i	{n,w}	{0,10}
	%-wi	%-10i	{n,-w}	{0,-10}
Вещественное	%f	%f	{n:f}	{0:f}
	%w.kf	%10.2f	{n:fk} {n,w:fk}	{0,f3} {0,15:f3}
	%-w.kf	%-10.2f	{n,-w:fk}	{0,-15:f3}
Строка	%s	%s	{n}	{0}
	%ws	%15s	{n,w}	{0,15}
	%-ws	%-15	{n,-w}	{0,-15}




```
%i  {0}
%10i  {0,10}
%-10i  {0,-10}
%8.3f  {0,10:f3}
%-8.3f  {0,-10:f3}
%10s  {0,10}
%-10s  {0,-10}
```

Структуры данных

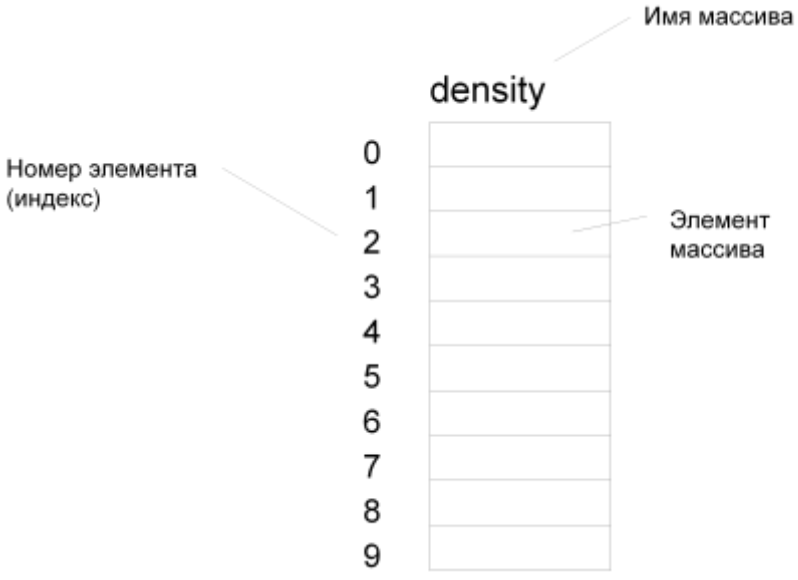
- Массив
- Список
- Файл
- Дерево
- Очередь
- Стек
- Словарь
- Коллекция

др.

Массивы

Массив – совокупность элементов ОДИНАКОВОГО типа
Массив:

- одномерный
- многомерный



имя[номер_элемента]

density[0]
density[i]
density[k+1]

Массив:

- статический
- динамический

C/C++ - статические и динамические
C# - только динамические

тип имя[размер];

```
double density[10];

#define K  10
double density[K];
```

тип[] *имя*;

```
double[] density;
density = new double[10];
int k = 10;
density = new double[k];
```

Ошибки при работе с массивами

Обращение к несуществующему элементу (выход индекса за допустимое значение)

Операции с массивами

- Ввод / вывод
- Обработка (сумма элементов, среднее арифметическое, кол-во элементов, удовлетворяющих условию, и т.д.)
- Поиск минимального (максимального элемента)
- Сортировка
- Поиск нужного элемента

Ввод / Вывод

```
// псевдокод !!!
a[k] double
for (int i=0; i < k; i++)
{
        write(i);   // подсказка
        read(a[i]);
}
```

```
for (int i=0; i < k; i++)
{
        write(a[i]);
}
```

Обработка массива

```
// псевдокод !!!
a[n] double // массив
sum double  // сумма положительных элементов
int positive // количество положительных элементов
mean double // среднее положительных элементов

for (int i=0; i < k; i++)
{
        if (a[i] > 0) then
        {
                sum = sum + a[i];
                positive = positive + 1;
        }
}
mean = sum / positive;
```

Минимальный элемент массива

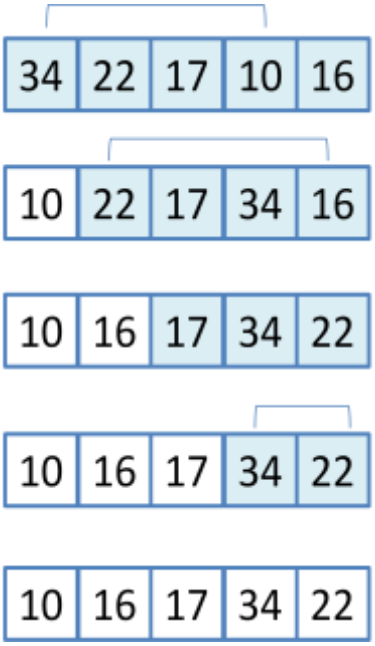
```
// псевдокод !!!
a[n] double // массив
int min     // номер минимального элемента

min = 0 // пусть первый элемент минимальный
for (int i=1; i < n; i++)
{
        if (a[i] < a[min] ) then
                min = i;
}
// здесь a[min]  - минимальный элемент массива
```

Сортировка массива

$a[0] \le a[1] \le a[2] \le \ldots \le a[k-1] \le a[k]$
$a[0] \ge a[1] \ge a[2] \ge \ldots \le a[k-1] \ge a[k]$

- Метод выбора
- Метод обменов (пузырька)
- Метод слияния
- ...

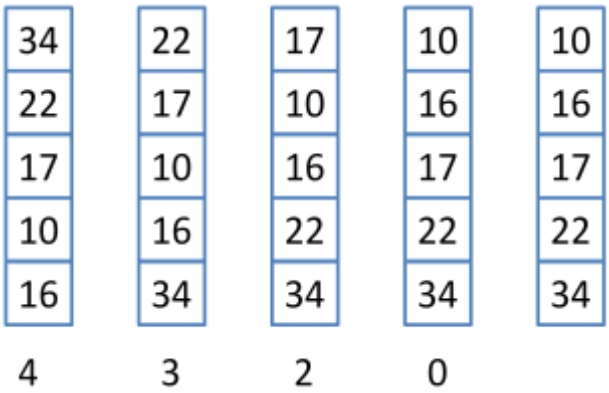
Метод перестановки



Метод перестановки

```
 for (int i = 0; i < k - 1; i++) {
   // просматривая массив с i-го эл-та найти min элемент
   int min = i;
   for (int j = i+1; j < k; j++) {
      if (a[j] < a[min])
           min = j;
    }
            // поменять местами i-й и min элементы
    b = a[i];
    a[i] = a[min];
    a[min] = b;
}
```

Метод "пузырька"




```
// Метод "пузырька"
for (int i = 0; i < n - 1; i++)  {
          for (int j = 0; j < n - 1; j++) {
```

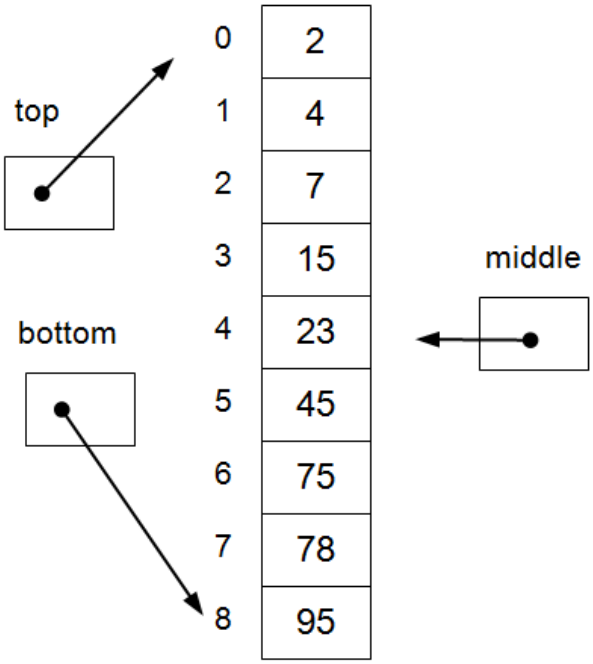
```
      if (a[j + 1] < a[j]) {
        // поменять местами j-й и j+1-й эл-ты
        b = a[j];
        a[j] = a[j + 1];
        a[j + 1] = b;
      }
  }
}
```

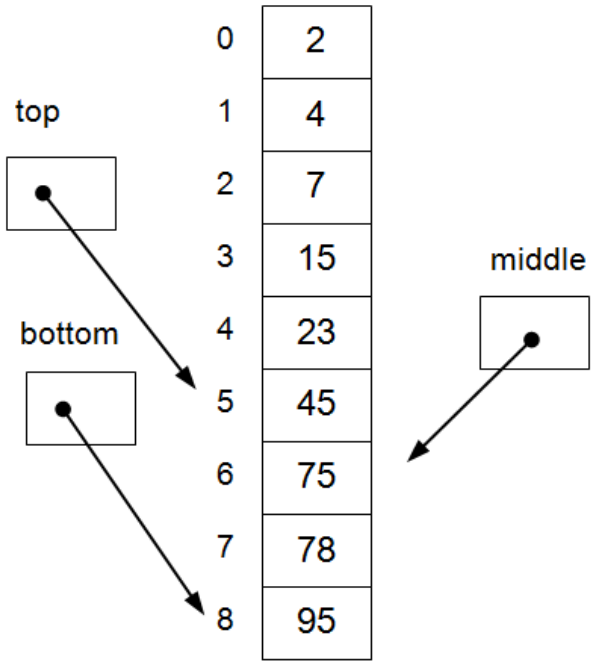
Поиск в массиве

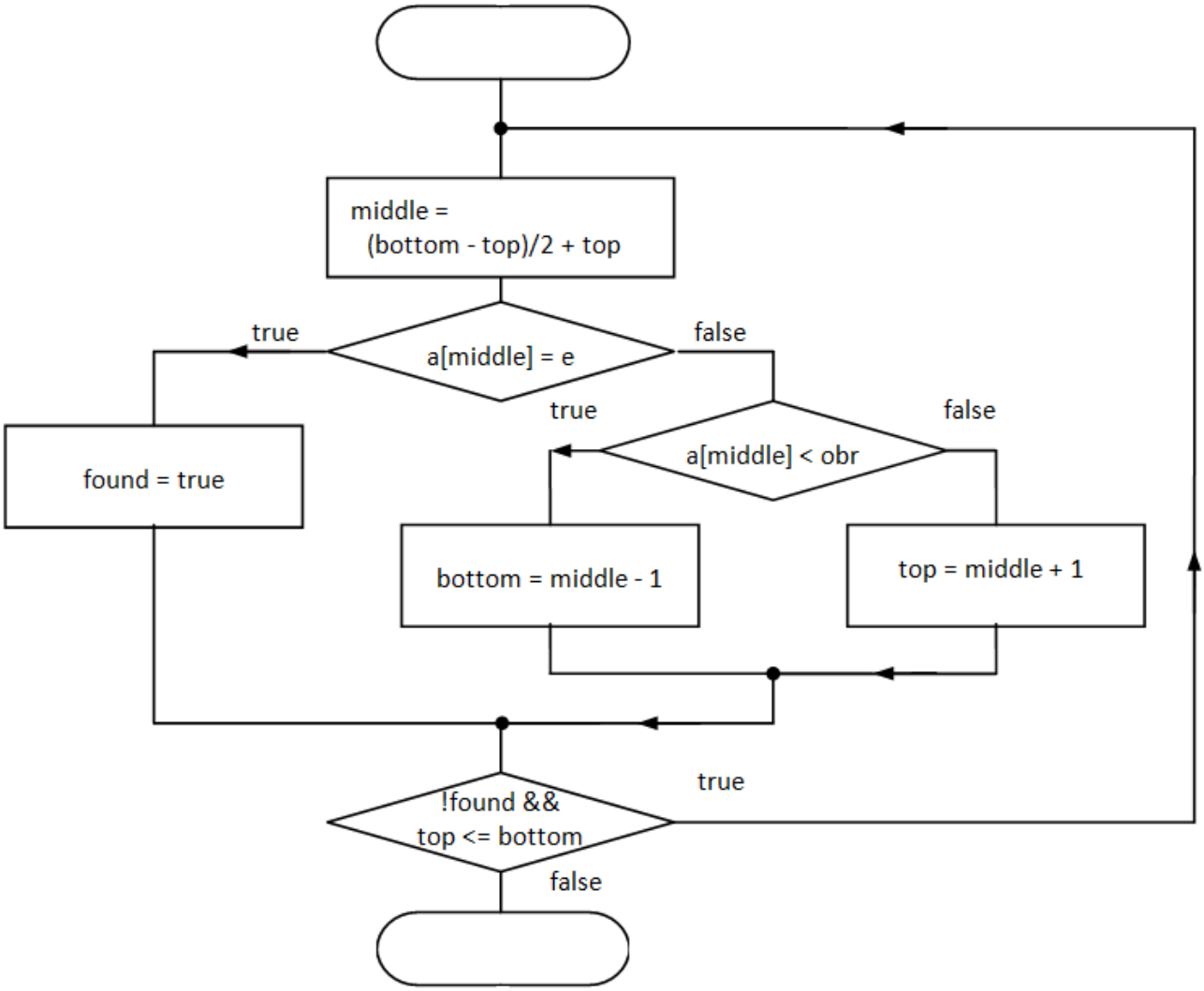
- Перебор
- Метод половинного деления

Результат – номер искомого элемента

Метод половинного деления








```
top = 0;         // верхняя граница
bottom = sz - 1;  // нижняя граница
middle; // средний (по номеру) элемент
found = false;

do {
    middle = (bottom - top) / 2 + top;
            if (a[middle] == e)
       found = true;
    else  {
       if (e < a[middle])
          // нужный элемент выше (до) middle
          bottom = middle - 1;
        else
          // нужный элемент ниже (после) middle
          top = middle + 1;
    }
 } while ( !found && top <= bottom);
```

Массив строк

```
string[] material  = {"Gold","Bronse","Silver"};

char mat01[] = {"Gold"};
char *material[] = {"Gold","Bronse","Silver"};
```

Двумерный массив

	янв	фев	мар		дек	всего
Chevrolet						
Ford						
Mazda						
Renault						
Всего						

```
C#
тип[,] имя;   // объявление
имя = new тип[строк, столбцов];  // создание
тип[] имя = new тип[строк, столбцов];

int[,] sales = new int[4, 13];

C/C++
тип имя[строк][столбцов]; // статический массив
int sales [4][13];
```

Ввод двумерного массива

```
// C#
for (int row = 0; row < nRow; row++)
   for (int col = 0; col < nCol; col++)
   {
      Console.Write("{0},{1}>", row, col);
      a[row, col] = Convert.ToInt32(Console.ReadLine());
   }
```

```
// C/C++
for (int row = 0; row < nRor; row++)
   for (int col = 0; col < nCol; col++)
   {
      printf("a[%i,%i] >", row, col);
      scanf("%i", &a[row][col]);
   }

// Псевдокод!
data[nRow,nCol] // массив данных
rowName[nRow]   // названия рядов данных
head[nCol]      // заголовки столбцов

for (int row = 0; row < nRow; row++)
{
   write(rowName[row]);
   for (int col = 0; col < nCol; col++)
   {
      write(head[col]);
      read(dada[row,col]);
   }
}
```

Сортировка двумерного массива

- Выбрать ключевое поле
- Менять местами СТРОКИ
- В качестве буферной переменной использовать дополнительную строку

Списки

Объявление

```
List<type> name;
name = new List<type>();

List<type> name = new List<type>();
```

Примеры

```
List<String> names = new List<Strings>();
List<Double> numbers = new List<Double>();
List<DateTime> dates = new List<DateTime>();
List<Persone> persones = new List<Persone>();
```

Создание списка

```
список.Add(объект);
```

```
names.Add("Иванов");
names.Add(name);
```

```
список.Insert(index,объект);
names.Insert(0,"Сидоров");
```

Вывод списка

```
Console.WriteLine(names[0]);
Console.WriteLine(names[i]);

int k = names.Count;
for (int i = 0; I < k; i++)
{
        Console.WriteLine(names[i]);
}
```

Поиск элемента в списке

```
список.IndexOf(Объект)

int p = names.IndexOf(name);
if ( p != -1)
{
        // элемент найден
}
```

Удаление элемента

```
список.RemoveAt(индекс);
список.Remove(объект);
```

Сортировка

```
список.Sort();
```

Выборка

```
List<string> menfolk = new List<string>();

for (int i = 0; i < names.Count; i++)
{
    String tail = names[i].Substring(names[i].Length-2);
    if (( tail == "ов") || (tail == "ин"))
    {
        menfolk.Add(names[i]);
    }
}
```

Подпрограмма

Подпрограмма - часть программы, обеспечивающая решение некоторой части общей задачи (подзадачи)

- Процедура
- Функция

Функция

- Стандартная (библиотечная)
- Программиста

Функция программиста

Функция - подпрограмма, выполняющая решение некоторой задачи

- Улучшение "читабельности" программы
- Повторное использование "удачного" кода
- Возможность параллельной разработки
- Возможность совершенствования частей программы

```
тип Имя (тип1 параметр1, тип2 параметр2, …)
{
   // объявление переменных

        // инструкции

   return значение;
}
```

Параметры

- для передачи инф. в функцию

- для возврата результата

```
//
// для доступа к sqrt и fabs добавить #include <math.h>

double CubeRoot(double x)
{
   double e = 0.1; // точность вычисления
   double p;       // текущее приближение

   p = sqrt(x); // первое приближение

   while ( fabs(p - x / (p * p)) > e)
   {
      p = (2 * p + x / (p * p)) / 3;
   }

   return p;
}

static double CubeRoot(double x)
{
   double e = 0.01; // точность вычисления
   double p;        // текущее приближение

   p = Math.Sqrt(x); // первое приближение

   while (Math.Abs(p - x / (p * p)) > e)
   {
      p = (2 * p + x / (p * p)) / 3;
   }

   return p;
}
```

Выполнение (вызов) функции

```
имя = выражение;

имя = функция(параметры);

cr = CubeRoot(100);
cr = CubeRoot(x);
cr = CubeRoot(a+b);
```

Использование функции

- В программу
- В модуль

- В библиотеку

```
// Использование функции – C/C++
double CubeRoot(double x) { ... }

int main()
{
   double cr = CubeRoot(x);
}

// Использование функции – C#
class Program
{
  static double CubeRoot(double x) {...}

  static void Main(string[] args)
  {
     double cr = CubeRoot(a);
  }
}
```

Перегрузка функций

```
static double CubeRoot(double x)
{
   double e = 0.01; // точность вычисления
   double p = Math.Sqrt(x); // первое приближение

   while (Math.Abs(p - x / (p * p)) > e)
      p = (2 * p + x / (p * p)) / 3;

   return p;
}

static double CubeRoot(double x, double e)
{
   double p = Math.Sqrt(x);

   while (Math.Abs(p - x / (p * p)) > e)
                  p = (2 * p + x / (p * p)) / 3;

   return p;
}

// Перегрузка функций - использование

cr = CubeRoor(100);
```

```
cr = CubeRoox(100, 0.0001);
```

void

void - пустой, пустота

```
void  Имя (тип1 параметр1, тип2 параметр2, …)
{
        // инструкции
}

// C#
static void myLine(char ch, int n)
{
   for (int i = 0; i<n; i++)
      Console.Write(ch);
    Console.WriteLine();
}


// C/C++
void myLine(char ch, int len)
{
  for (int i = 1; i < len; i++)
     putchar(ch);
  putchar((int) '\n');
}


myLine('=', n);
myLine('-', 25);
myLine('=', 25);
```

Переменные

- локальные
- внешние (глобальные, общие)

Рекурсия

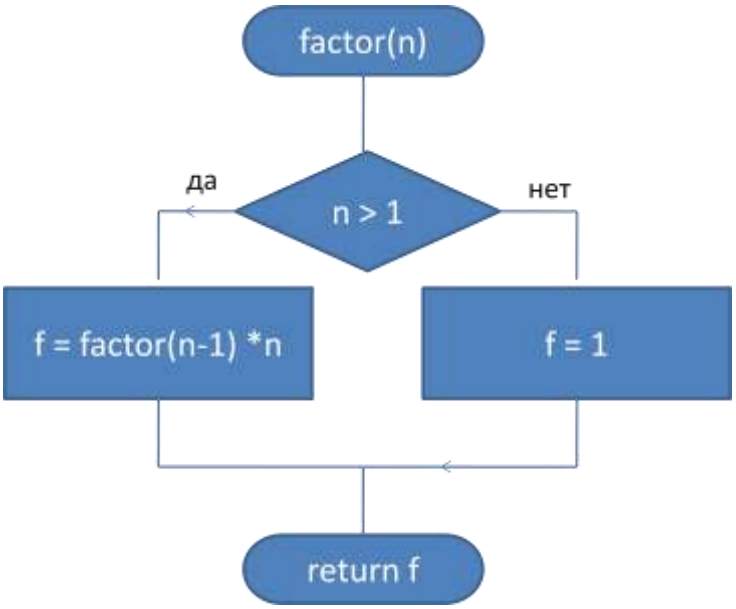
Объект называется рекурсивным, если он частично или полностью состоит из объектов (состоит) этого типа, экземпляром которого он является.

- Факториал числа n это - произведение числа n на факториал n.
- Каталог - это объект, который состоит из (содержит) файлы и каталоги.

Факториал

```
 0 1
 1 1
 2 2
 3 6
 4 24
 5 120
 6 720
 7 5040
 8 40320
 9 362880
 10 3628800
 11 39916800
 12 479001600
 13 6227020800
 14 87178291200
 15 1307674368000
```

n! = 1*2*3* ...*(n-1)*n
n! = ((n-2)!(n-1))*n



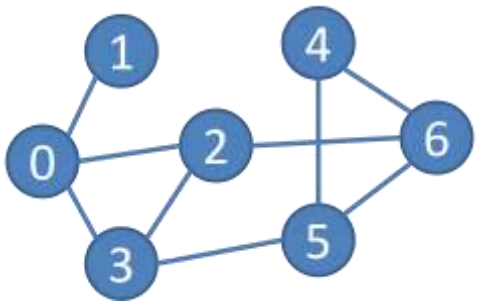

```
// C#
static Int64 factorial(int n)
{
   Int64 f;
   if (n > 1)
     f = factorial(n - 1) * n;
   else f = 1;

   return f;
}
```

```
Int64 f;
for (int n = 0; n <= 15; n++)
{
    f = factorial(n);
    Console.WriteLine(" {0} {1}", n, f);
}
```

```
// C/C++
__int64 factorial(__int64 n)
{
   __int64 f;
   if (n > 1)
     f = factorial(n - 1) * n;
   else f = 1;
   return f;
}

for (long n = 0; n <= 15; n++)
{
   __int64 f = factorial(n);
   printf(" %i %I64u\n", n,  f);
}
```

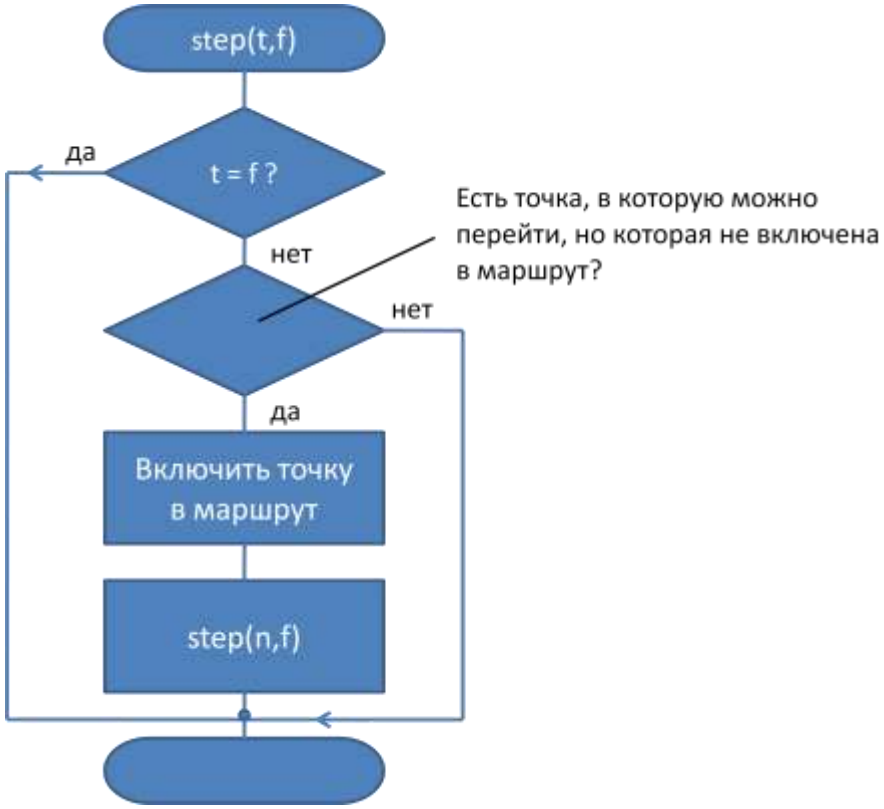
Поиск пути на графе

Граф - совокупность узлов и дуг; множество вершин и ребер



	0	1	2	3	4	5	6
0	0	1	1	1	0	0	0
1	1	0	0	0	0	0	0
2	1	0	0	1	0	0	1
3	1	0	1	0	0	1	0
4	0	0	0	0	0	1	1
5	0	0	0	1	1	0	1

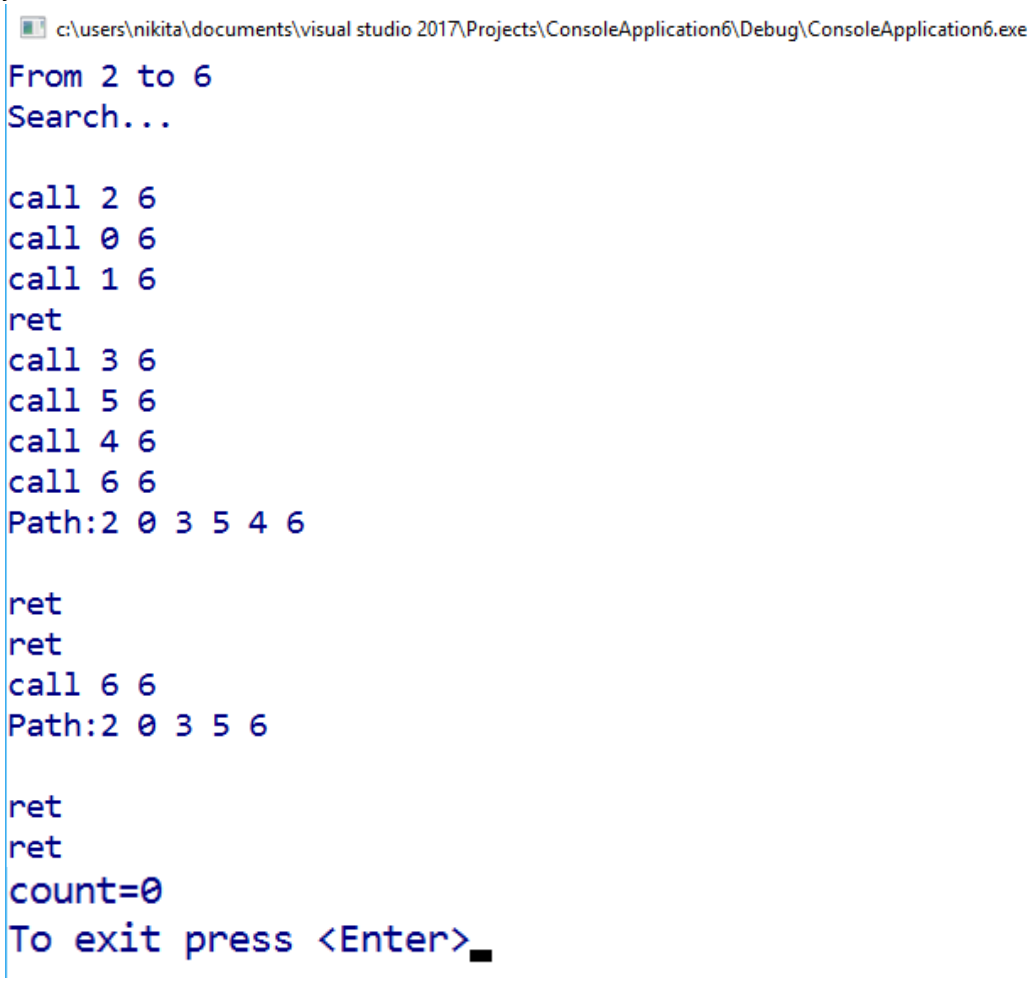
6	0	0	1	0	1	1	0




```
// поиск маршрута
step(0,6)

Поиск маршрута (пути)
// ПСЕВДОКОД !
// map[,] - карта; road[] - путь;
// incl[k] == 1, если точка k включена в маршрут

//  s - откуда, f - куда, p - номер точки маршрута
void step(s,f,p) {
   if (s == f)  return;
   for (c=0; c < N; c++) {
      if ( ( map[s,c] !=0) and ( ! incl[c])) {
           road[p] = c;
           incl[c] = true;
           step(c,f,p+1);

          // здесь путь найден или точка тупиковая

          road[p] = 0;
          include[c] = false;
     }
    }
}
```

```
void step(int s, int f, int p)
{
   // count - внешний сч-к вызовов/возвратов
   count++;
   printf("\ncall %i %i", s, f);

   if (s == f) {
      printf("\nPath:");
      for (int i = 0; i < N; i++)
         if ( road[i] != -1)
            printf("%i ", road[i]);
      printf("\n");
   }


  else
   for (int c = 0; c < N ; c++) {
     if ((map[s][c] != 0) && (inRoad[c] == 0)) {
         road[p] = c;
         inRoad[c] = 1;

      step(c, f, p + 1);

      road[p] = -1;
      inRoad[c] = 0;
     }
   }
  count--;
  printf("\nret");
}
int map[N][N];
 int road[N];
int inRoad[N];
int count = 0;

void step(int s, int f, int p) { ... }

int main()
{
   int start = 2;
   int finish = 6;

   road[0] = start;
   inRoad[start] = 1;

   step(start, finish, 1);

   printf("\ncount=%i", count);
```

```
    return 0;
}
```

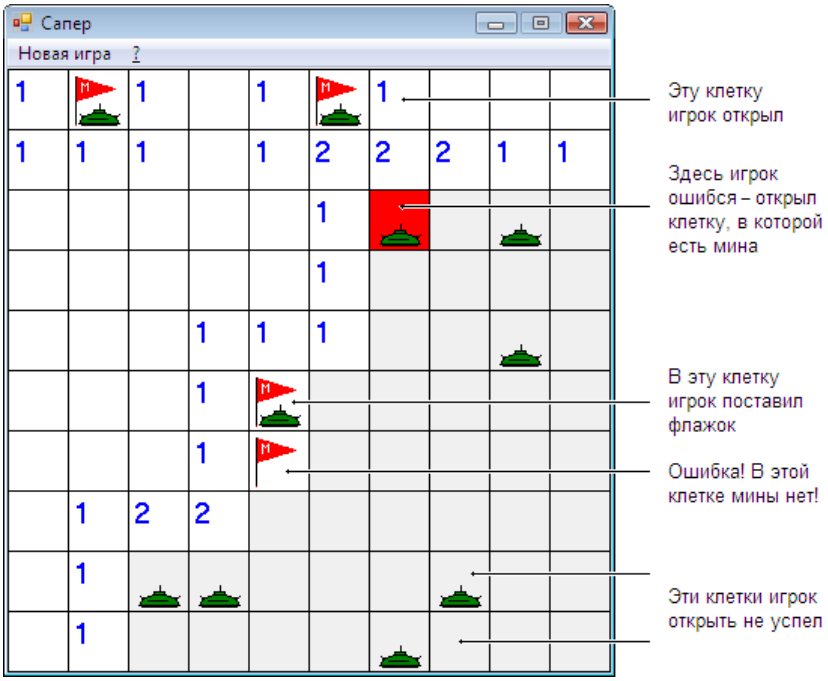


Очистка (обработка) диска

```
void Clear() {
   // обработать файлы текущего каталога

   // получить список каталогов

   foreach aDir in DirList {
      if (aDir != "..")  && (aDir != ".") {
         ChDir(aDir);  // войти в каталог
         Clear();
         ChDir("..");  // выйти из текущего каталога
      }
    }
}

// очистка диска
ChDir(aDir);
```

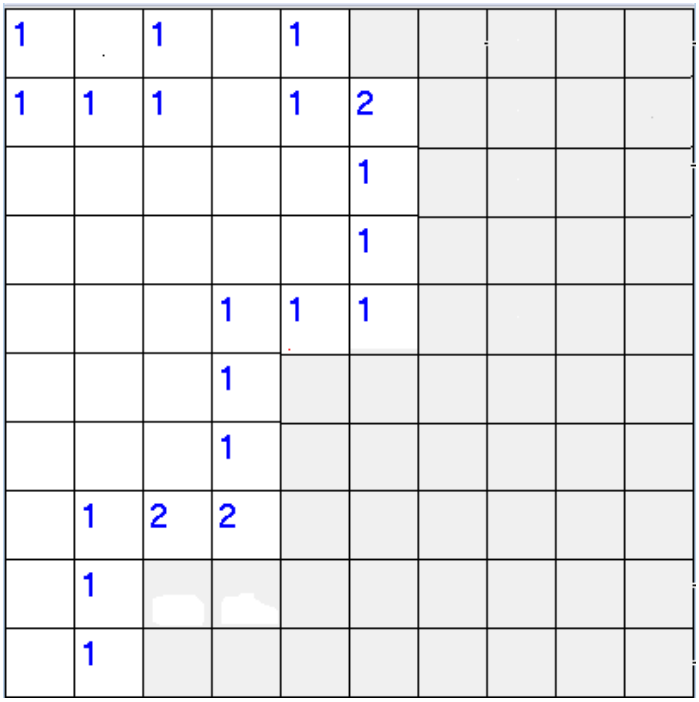
Clear();

Рекурсия – игра Сапер



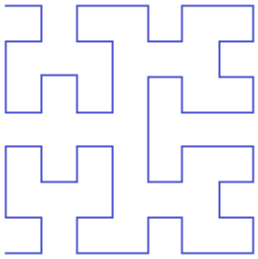
–3	–3	–3	–3	–3	–3	–3	–3	–3	–3	–3	–3
–3	**9**	1	0	0	0	0	0	0	0	0	–3
–3	1	1	0	0	0	0	0	0	0	0	–3
–3	1	2	2	1	0	0	0	1	1	1	–3
–3	1	**9**	**9**	1	0	0	0	2	**9**	2	–3
–3	1	2	2	1	0	0	0	2	**9**	3	–3
–3	0	0	0	0	0	0	0	2	3	**9**	–3
–3	0	1	2	2	1	0	0	1	**9**	2	–3
–3	0	2	**9**	**9**	1	0	0	1	1	1	–3
–3	0	2	**9**	3	1	0	0	0	0	0	–3
–3	0	1	1	1	0	0	0	0	0	0	–3
–3	–3	–3	–3	–3	–3	–3	–3	–3	–3	–3	–3

0-8 количество мин в соседних клетках
9 - мина
+ 100 - поставлен флажок
+200 - клетка открыта
-3 - граница поля




```
void open(int row, int col)
{
   if (Pole[row,col] == 0)    {
      Pole[row,col] = 100;

      kletka(g, row, col, status); // отобразить содержимое клетки

      // открыть примыкающие клетки слева, справа, сверху, снизу
      open(row, col-1);
      open(row-1, col);
      open(row, col+1);
      open(row+1, col);

      //примыкающие диагонально
      open(row-1,col-1);
      open(row-1,col+1);
      open(row+1,col-1);
      open(row+1,col+1);
   }
   else
      if ((Pole[row,col] < 100) &&   (Pole[row,col] != -3))  {
         Pole[row,col] += 100;
          kletka(g, row, col, status); // отобразить содержимое клетки
      }
}
```

Кривые Гильберта



Внимание , рекурсия!

- Параметры рекурсивной функции сохраняются в стеке, размер которого ОГРАНИЧЕН!
- Возможно ИСКЛЮЧЕНИЕ "Переполнение стека"
- Использовать для работы с объектами и задачами, которые по своей природе рекурсивны

Введение в ООП

Процедурное программирование

"Определите, какие процедуры вам нужны; используйте лучшие алгоритмы, которые только сможете найти"

Объектно-ориентированное программирование

"Определите, какие классы вам нужны; заготовьте полный набор операций для каждого класса; выразите общие свойства явным образом, используйте наследование"

Класс - это сложная структура данных, определяемая пользователем (Дьюхарст С., Старк К.)
Класс - это определяемый пользователем тип (Страуструп Б.)

Задача: написать программу вычисления массы полого стержня.

Процедурный подход:
формула (алгоритм) + исходные данные

m = volume * density
len, r1, r2, density

m = 3.14*(r12 - r22)*len *density

ООП подход
Свойство - характеристика объекта

Стержень:
длина
диаметр
диаметр отверстия
плотность материала
объем
масса

Объявление класса

```
// C++
class Имя
{
        // данные (поля)

  // свойства

  // конструктор

  // деструктор

  // методы

}

// C#
class Имя
{
        // данные (поля)
  // свойства
  // конструктор
  // методы
}


// C++
class TRod // стержень
{
   double _ext_diameter;  // внешний диаметр
   double _hole_diameter; // диаметр отверстия
   double _length;       // длина
   double _density;      // плотность материала
```

```
public:
   TRod(double ext_diameter, double hole_diameter,
         double length, double density) { ... }

    TRod(double diameter, double length, double density){ ... }

    //
    double volume() { ... }
    double weight() { ... }

    void setLength (double length) { ... }
    double getLength () { ...}

}
TRod(double ext_diameter, double hole_diameter, double length, double density) {
   _ext_diameter = ext_diameter;
   _hole_diameter = hole_diameter;
   _length = length;
   _density = density;
}

TRod(double diameter, double length, double density) {
_ext_diameter = diameter;
_hole_diameter = 0;
_length = length;
_density = density;

}
double volume() {
   return 3.1415926 * (_ext_diameter * _ext_diameter - _hole_diameter* _hole_diameter) /
4000000000;
}

double weight() {
   return this->volume() * _density;
}

int main()
{

   TRod* rod1 = new TRod(736, 1000, 2700);

   printf("Volume: %f\nWeight: %f\n", rod1->volume(),
          rod1->weight());

   printf("\nTo exit press <Enter>");
   int ch = getchar();

   return 0;
```

```
}

// C#
class  TRod {
   int d1;     // внешний диаметр
   int d2;     // диаметр отверстия
   int len;    // длина
   double dens; // плотность материала

   public TRod() { d1 = 0; d2 = 0; len = 0; dens = 0; }

   public TRod(int ext_diameter, int hole_diameter, int length,
               double density)    {
      d1 = ext_diameter;
      d2 = hole_diameter;
      len = length;
      dens = density;
   }

   public TRod(int ext_diameter, int length, double density) { ... }

   public int Diametr { get { return d1; } set { d1 = value; } }
   public int Hole { get { return d2; } set { d2 = value; } }
   public int Length { get { return len; } set { len = value; } }
   public double Densyty { get { return dens; } set { dens = value; } }

   public double Volume { get { return len * Math.PI *
               Math.Pow((d1 - d2)/2, 2) / ( Math.Pow(10, 9)); } }
   public double Mass { get { return this.Volume * dens; } }
}
static void Main(string[] args)
{
   TRod rod = new TRod(25, 1500, 520); // сосна

   TRod rod2 = new TRod();
    rod2.Length = 1500;
    rod2.Diametr = 25;
    rod2.Hole = 20;
    rod2.Densyty = 2700; // алюминиевый сплав

    Console.WriteLine("Объем: {0:f6} куб.м, Масса: {1:f3} кг", rod.Volume, rod.Mass);
    Console.WriteLine("Объем: {0:f6} куб.м, Масса: {1:f3} кг", rod2.Volume, rod2.Mass);

    Console.Write("To exit press any key ...");
    Console.ReadKey();
}
```

Список объектов

```
class Sale
{
    String ftitle;
    int fvolume;

    public String Title { get { return ftitle; } set { ftitle = value; } }
    public int Volume { get { return fvolume; } set { fvolume = value; } }

    public Sale(String title, Int32 volume)
    {
        ftitle = title;
        fvolume = volume;
    }
}

static void Main(string[] args)
{
    List<Sale> sales = new List<Sale>();

    sales.Add(new Sale("Toyota", 15400));
    sales.Add(new Sale("Mazda", 12700));
    sales.Add(new Sale("Ford", 24500));
    sales.Add(new Sale("Renault", 48300));
    sales.Add(new Sale("Hunday", 35200));

    for (int i = 0; i < sales.Count; i++)
    {
        Console.WriteLine("{0} - {1}", sales[i].Title, sales[i].Volume);
    }
}
```

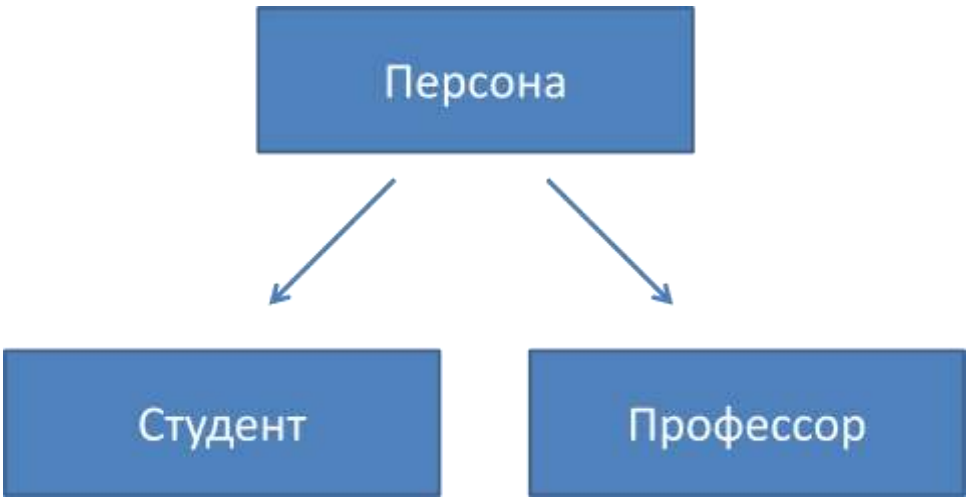
Принципы ООП

- Наследование
- Полиморфизм
- Виртуальные функции (методы)

Студент это - **персона**, которая учиться в университете.
Преподаватель это - **персона**, которая дает (передает) знания студенту.

Профессор это - **преподаватель**, имеющий ученую степень доктора наук.
Доцент это - **преподаватель**, имеющий степень кандидата технических наук

Наследование



Персона : имя, фамилия, дата рождения
Студент: имя, фамилия, дата рождения, уч. группа
Преподаватель: имя, фамилия, дата рождения, кафедра, уч. степень, науч. звание

Персона : имя, фамилия, дата рождения
Студент: персона + уч. группа
Преподаватель: персона + кафедра, уч. степень

Дочерний класс может иметь доступ к свойствам и методам родительского класса.

Базовый (родительский) класс
Производный (дочерний) класс

```
class TPersona {
      string fname;
      string lname;

      public TPersona(string FirstName, string LastName) {
            fname = FirstName;
            lname = LastName;
          }

      public TPersona() { fname = ""; lname = ""; }

      public string FirstName { get { return fname; } set { fname = value; } }
      public string LastName { get { return lname; } set { lname = value; } }
      public string FullName { get { return fname + " " + lname; } }
   }

 TPersona[] simpsons = new TPersona[5];

 simpsons[0] = new TPersona("Bart", "Simpson");
 simpsons[1] = new TPersona("Homer", "Simpson");

 for (int i =0; i < 5; i ++) {
```

```
    if (simpsons[i] != null)
        Console.WriteLine("{0}", simpsons[i].FullName);
 }
```

Производный класс

```
class имя : базовый_класс
{
        // поля
        // конструкторы
        // свойства
        // методы
}

class TStudent : TPersona {
       int group;

       public TStudent(string FirstName, string LastName, int Group)
       {
          this.FirstName = FirstName;
          this.LastName = LastName;
          group = Group;
       }

       public int Group { get { return group; } set { group = value; } }
}

TStudent[] students = new TStudent[10];

students[0] = new TStudent("Барт", "Симпсон", 1358);
students[1] = new TStudent("Милхаус", "Ван Хутен", 1358);

for (int i = 0; i < 10; i++) {
    if (students[i] != null)
        Console.WriteLine("{0}", students[i].FullName);
}

class TProf : TPersona {
       string kafedra;

       public TProf(string FirstName, string LastName, string Kafedra)
       {
          this.FirstName = FirstName;
          this.LastName = LastName;
          kafedra = Kafedra;
       }

       public string Kafedra { get { return kafedra; }
                           set { kafedra = value; } }
```

```
 }

 TProf[] profs = new TProf[10];

profs[0] = new TProf("Гомер", "Симпсон", "Энергетические системы");
profs[1] = new TProf("Скрудж", "Макдак", "Мировая экономика");

 for (int i = 0; i < 10; i++) {
   if (profs[i] != null)
     Console.WriteLine("{0}, {1}", profs[i].FullName, profs[i].Kafedra);
 }
TPersona[] persones = new TPersona[10];

persones[0] = new TStudent("Барт", "Симпсон", 1358);
persones[1] = new TStudent("Милхаус", "Ван Хутен", 1358);
persones[2] = new TProf("Гомер", "Симпсон", "Энергетические системы");
persones[3] = new TProf("Скрудж", "Макдак", "Мировая экономика");

Console.WriteLine("Persones:");
for (int i = 0; i < 10; i++) {
    if (persones[i] != null)
       Console.WriteLine(persones[i].FullName);
}

Console.WriteLine("\nStudents:");
for (int i = 0; i < 10; i++)  {
    if ( (persones[i] != null) && (persones[i] is TStudent))
        Console.WriteLine(persones[i].FullName);
}
```

Полиморфизм и виртуальные функции

Полиморфизм - способность функции обрабатывать данные разных типов.

Info() - выводит информацию о студенте или преподавателе, в зависимости от того, к какому объекту применен метод

```
class TStudent : TPersona {
      int group;
      public TStudent(string FirstName, string LastName, int Group)  ...
      public int Group  ...

      public string Info()
      {
         return this.FullName + ", гр. " + this.group.ToString();
      }
```

```
}

class TProf : TPersona {
      string kafedra;
      public TProf(string FirstName, string LastName, string Kafedra ...
      public string Kafedra { ... }

      public string Info()
      {
         return this.FullName + ", каф. " + this.kafedra;
      }
   }
TPersona[] persones = new TPersona[10];

// BAD !!!
j = 0;
while ((j < 10) && (persones[j] != null)) {
     if ( persones[i] is TProf)
          WriteLine( (persones[j] as TProf).Info());
      else
          WriteLine((persones[j] as TStudent).Info());
      j++;
}

// Cool (Ottimo, ma ...)
// чтобы это работало, надо в базовом классе определить метод Info
// и переопределить его в дочерних классах
j = 0;
while ( (j < 10) && (persones[j] != null) ) {
      WriteLine(persones[j].Info());
      j++;
}
```

Наследование – C# реализация

```
class TPersona
{
      string fname;
      string lname;

      public TPersona(string FirstName, string LastName) ...
      public TPersona() ...

      public string FirstName  ...
      public string LastName  ...
      public string FullName  ...
```

```
    public virtual string Info() { return "";  }
}


class TStudent : TPersona {
    int group;
    public TStudent(string FirstName, string LastName, int Group)  ...
    public int Group  ...

    public override string Info()
    {
      return this.FullName + ", гр. " + this.group.ToString();
    }
}

class TProf : TPersona {
    string kafedra;
    public TProf(string FirstName, string LastName, string Kafedra ...
    public string Kafedra  ...

    public override string Info()
    {
      return this.FullName + ", каф. " + this.Kafedra;
    }
}
```

Наследование – C++ реализация

```
class TPerson
{
  char *fname;
  char *lname;

public:
  TPerson() {} // чтобы иметь возможность создать массив объектов

  TPerson(char* FirstName, char* LastName)
  {
    fname = new char[strlen(FirstName) + 1]; strcpy(fname, FirstName);
    lname = new char[strlen(LastName) + 1];strcpy(lname, LastName);
  }

  ~TPerson()
  {
    delete[] fname;
    delete[] lname;
  }

  char* FullName()
```

```
   {
      char* fullname = new char[strlen(fname) + strlen(lname) + 2];
      sprintf(fullname, "%s %s", fname, lname);
      return fullname;
   };

   virtual void print()  { }

};
// студент
class TStud : public TPerson
{
   int group;
public:
   TStud() {};

   TStud(char* FirstName, char* LastName, int Group):TPerson(FirstName,LastName)
   {
      group = Group;
   }

   ~TStud()
   {
   }

   void print()    // выводит информацию о студенте
   {
      printf("%s, gr. %i\n", this->FullName(), group);
   }
};

// профессор
class TProf : public TPerson
{
   char* kafedra;

public:
   TProf() {};

   TProf(char* FirstName, char* LastName, char* Kafedra)
                              :TPerson(FirstName, LastName)
   {
      kafedra =
         new char[strlen(Kafedra) + 1]; strcpy(kafedra, Kafedra);
   }

   ~TProf()
   {
      delete[] kafedra;
```

```
    }

    void  print()    // выводит информацию о профессоре
    {
        printf("%s, dep. %s\n", this->FullName(), kafedra);
    }

};
int main()
{
    TPerson *pers[10]; //  студенты и преподаватели

    // Внимание! Значение, возвращаемое sizeof зависит от
    // разрядности платформы!!
    int K = sizeof(pers) / sizeof(TPerson*);

    // инициализация массива ОБЯЗАТЕЛЬНА
    for (int i = 0; i < K; i++)
        pers[i] = NULL;

    pers[0] = new TStud("Bart", "Simpson", 1358);
    pers[1] = new TStud("Lisa", "Simpson", 1358);
    pers[2] = new TProf("Homer", "Simpson", "Nucler Energi Systems");

    int i = 0;
    while (pers[i] != NULL)
    {
        pers[i]->print();
        i++;
     }

    // продолжение main

    // чтобы typid стала доступна, надо добавить #include <typeinfo>

    printf("\nStudents:\n");
    i = 0;
    while (pers[i] != NULL)
    {
        if (typeid(*pers[i]) == typeid(TStud)) {
            pers[i]->print();
        }
        i++;
    }

    // освободить память!


    printf("\nTo exit press <Enter>");
```

```
   int ch = getchar();

   return 0;
}
```

Ошибки при работе с объектами

Обращение к несуществующему объекту - System.NullReferenceException

Вывод - Перед обращением к объекту проверить, что он не null (NULL)!

Файлы

Файл

- Текстовый
- Двоичный

Потоки

- чтения
- записи

Поток чтения:
System.IO.StreamReader

Поток записи:
System.IO.StreamWriter

Чтение из файла

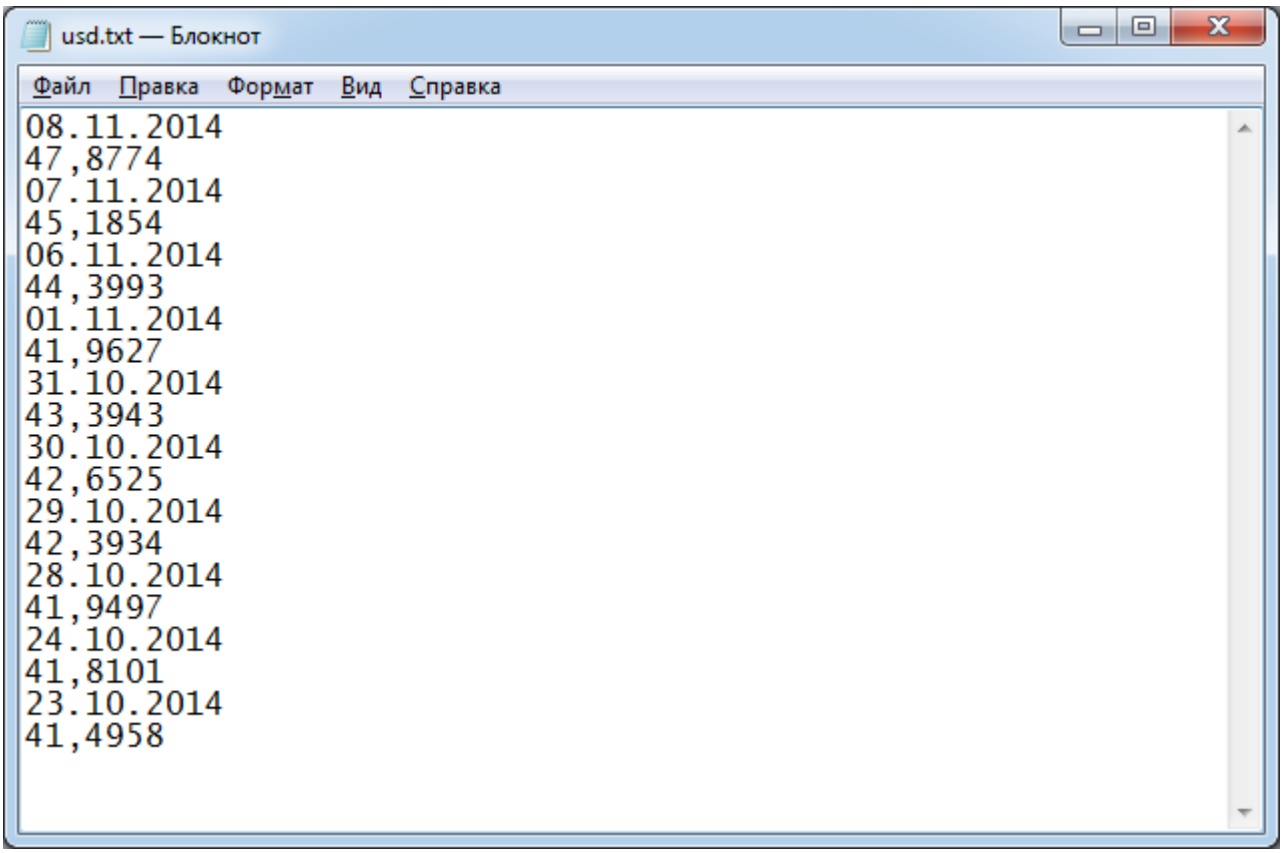
```
System.IO.StreamReader sr;
sr = new System.IO.StreamReader(файл);
```

Файл: Путь\имя_файла

Application.StatrupPath – путь к папке, из которой запущена программа

Папки компьютера

```
Environment.GetFolderPath(Environment.SpecialFolder.Desktop)
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
Environment.GetFolderPath(Environment.SpecialFolder.MyPictures)
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)
```




```
path = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

string textFile = path + Path.DirectorySeparatorChar + "usd.txt";

if (  System.IO.File.Exists(textFile))
{
   System.IO.StreamReader sr = new StreamReader(textFile);
   while ( ! sr.EndOfStream)
   {
       string st = sr.ReadLine();
       string st2 = sr.ReadLine();
       Console.WriteLine("{0} {1}",st,st2);
   }
   sr.Close();
}
else
{
   Console.WriteLine("Нет файла данных " + textFile);
}
```

Запись в файл

System.IO.StreamWrirer

sw = new System.IO.StreamReader(файл, mode);

mode:

- true – добавление
- false - перезапись

```
bool mode = true; //  true - добавить; false - перезаписать
System.IO.StreamWriter sw = new StreamWriter(textFile, mode);

string st3;
int k = 0;
do
{
    Console.Write(">>");
    st3 = Console.ReadLine();
     if ( st3.Length !=0)
     {
        sw.WriteLine(st3);
        k++;
     }
}
while (st3.Length != 0);
sw.Close();
```

Отладка

Этапы разработки программы:

- постановка задачи
- проектирование
- кодирование
- отладка и тестирование
- сопровождение

Ошибки

- синтаксические
- алгоритмические

Тест – набор исходных данных и соответствующий им набор выходных данных

Пример
Расчет массы полого стержня
Исходные данные: длина, внешний диаметр, плотность
Исходные значения: ???

Исходные значения: длина - 1000 мм, внешний диаметр - 1128 мм
Результат: масса численно равна плотности материала - ОК!

Методы отладки

- отладочная печать
- пошаговое выполнение

Отладочная печать

```
static void Main(string[] args)
{
    int a  = 0;
    int b = 0;
    int c= 0;

    System.Console.Write("a>");
    a = System.Convert.ToInt32(Console.ReadLine());

    System.Console.Write("b>");
    a = System.Convert.ToInt32(Console.ReadLine());


    // отладочная печать
    System.Console.WriteLine("a={0} b={1}", a, b);

    c = a + b;

    System.Console.WriteLine("c={0}",c);

    System.Console.ReadKey();

}
```

Условная компиляция

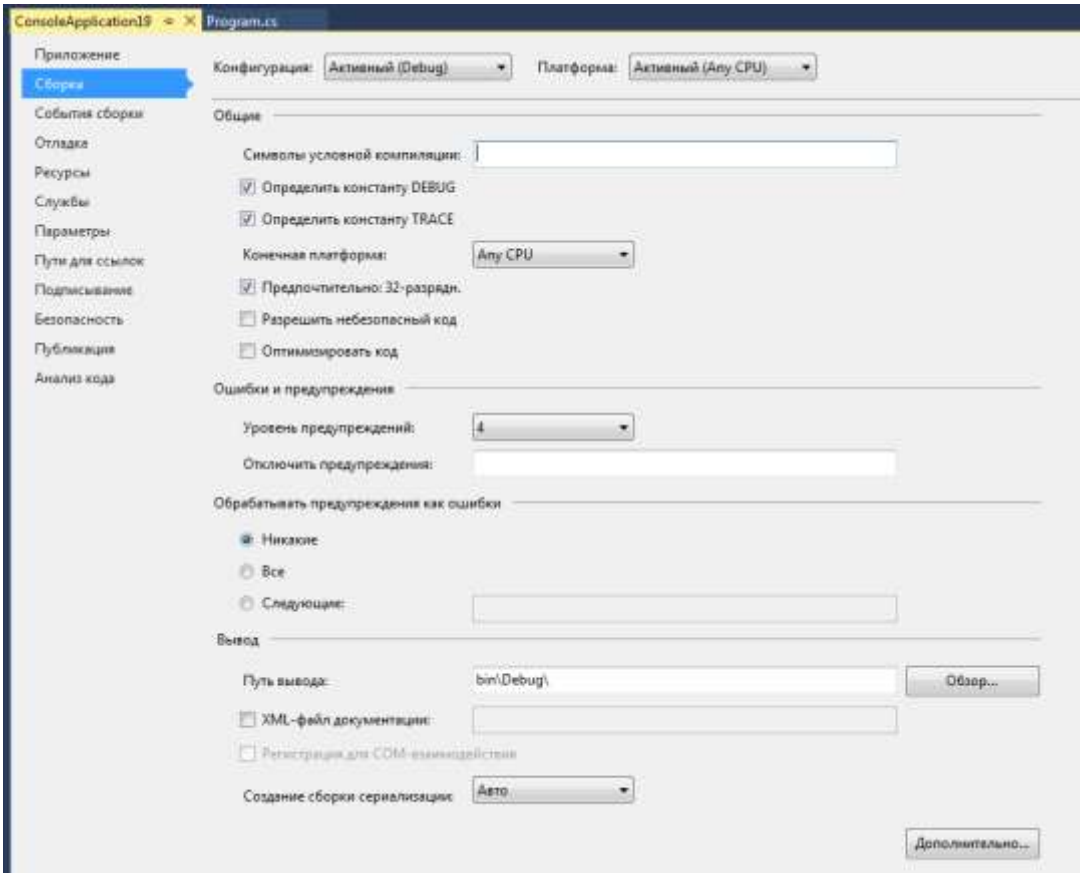
Директивы препроцессору

#define ИМЯ

#if ИМЯ
// здесь инструкции программы
#endif

```
#define DEB

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int a  = 0;
            int b = 0;
            int c= 0;

            System.Console.Write("a>");
            a = System.Convert.ToInt32(Console.ReadLine());

            System.Console.Write("b>");
            a = System.Convert.ToInt32(Console.ReadLine());

#if DEB
            System.Console.WriteLine("a={0} b={1}", a, b);
#endif
            c = a + b;

            System.Console.WriteLine("c={0}",c);

            System.Console.ReadKey();

        }
    }
}
```

```
//#define DEB

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int a  = 0;
            int b = 0;
            int c= 0;

            System.Console.Write("a>");
            a = System.Convert.ToInt32(Console.ReadLine());

            System.Console.Write("b>");
            a = System.Convert.ToInt32(Console.ReadLine());

#if DEB
            System.Console.WriteLine("a={0} b={1}", a, b);
#endif
            c = a + b;

            System.Console.WriteLine("c={0}",c);

            System.Console.ReadKey();

        }
    }
}
```


ConsoleApplication19
Program.cs
Приложение
Сборка
События сборки
Отладка
Ресурсы
Службы
Параметры
Пути для ссылок
Подписывание
Безопасность
Публикация
Анализ кода
Конфигурация: Активный (Debug)
Платформа: Активный (Any CPU)
Общие
Символы условной компиляции:
Определить константу DEBUG
Определить константу TRACE
Конечная платформа: Any CPU
Предпочтительно: 32-разряд.
Разрешить небезопасный код
Оптимизировать код
Ошибки и предупреждения
Уровень предупреждений: 4
Отключить предупреждения:
Обрабатывать предупреждения как ошибки
Никакие
Все
Следующие:
Вывод
Путь вывода: bin\Debug\
Обзор...
XML-файл документации:
Регистрация для COM-взаимодействия
Создание сборки сериализации: Авто
Дополнительно...

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int a  = 0;
            int b = 0;
            int c= 0;

            System.Console.Write("a>");
            a = System.Convert.ToInt32(Console.ReadLine());

            System.Console.Write("b>");
            a = System.Convert.ToInt32(Console.ReadLine());

#if DEBUG
            System.Console.WriteLine("a={0} b={1}", a, b);
#endif
            c = a + b;

            System.Console.WriteLine("c={0}",c);

            System.Console.ReadKey();

        }
    }
}
```

Пошаговое выполнение

Точка останова

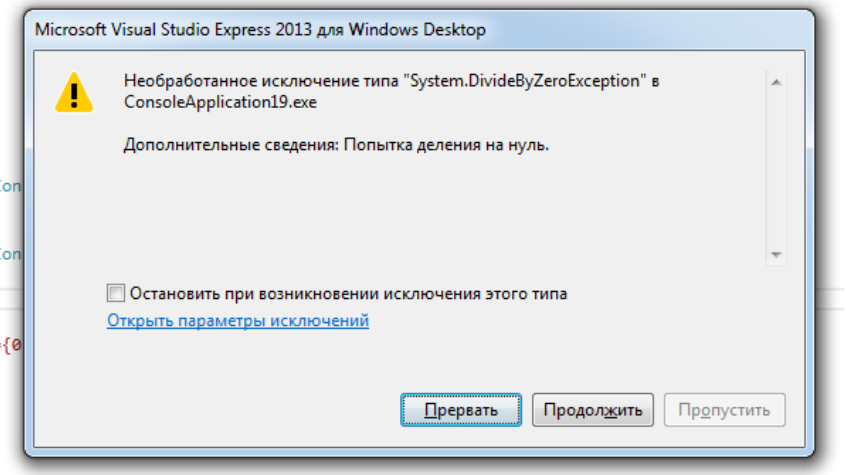
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int a  = 0;
            int b = 0;
            int c= 0;

            System.Console.Write("a>");
            a = System.Convert.ToInt32(Console.ReadLine());

            System.Console.Write("b>");
            a = System.Convert.ToInt32(Console.ReadLine());

            c = a + b;

            System.Console.WriteLine("c={0}",c);

            System.Console.ReadKey();

        }
    }
}
```

Продолжение выполнения

- Отладка>Продолжить (<F5>)
- Отладка> Шаг с обходом (<F10>)
- Отладка> Шаг с заходом (<F11>)
- Остановить отладку (<Shift>+<F5>)

Исключение

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int a  = 0;
            int b = 0;
            int c= 0;

            System.Console.Write("a>");
            a = System.Convert.ToInt32(Con

            System.Console.Write("b>");
            a = System.Convert.ToInt32(Con

            c = a / b;

            System.Console.WriteLine("c={0

            System.Console.ReadKey();

        }
    }
}
```

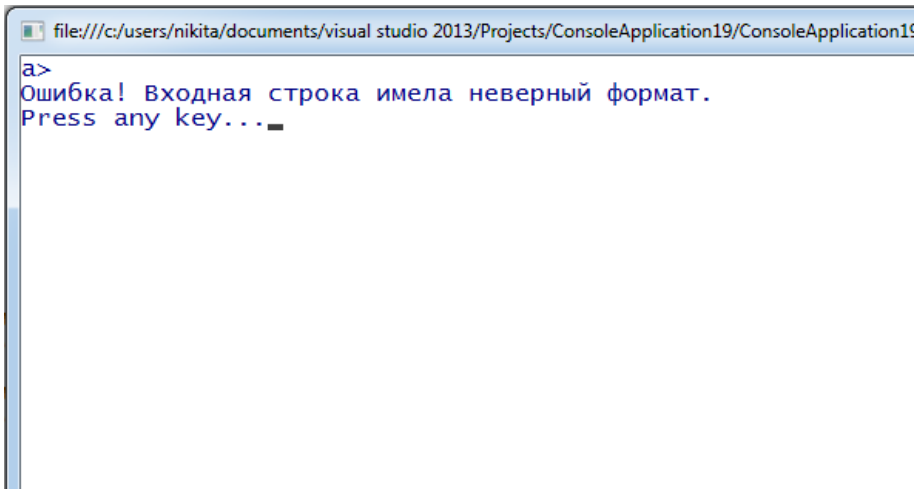


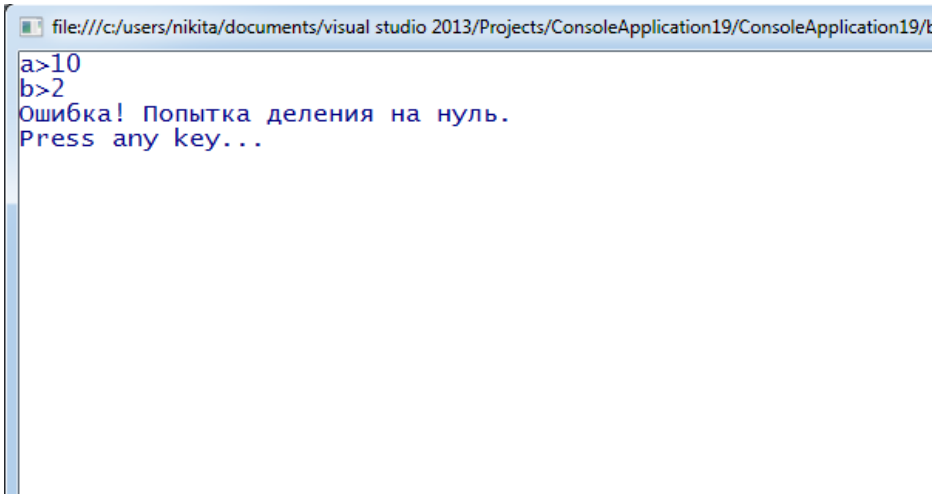
Исключения

- Systen.FormatException
- System.DevideByZeroException
- System.IndexOutOfRangeException
- System.NullReferenceException

Обработка исключения

```
namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int a  = 0;
            int b = 0;
            int c= 0;

            try
            {

                System.Console.Write("a>");
                a = System.Convert.ToInt32(Console.ReadLine());

                System.Console.Write("b>");
                a = System.Convert.ToInt32(Console.ReadLine());

                c = a / b;

                System.Console.WriteLine("c={0}",c);

            }

            catch (Exception aException)
            {
                Console.WriteLine("Ошибка! {0}", aException.Message);
            }

            System.Console.Write("Press any key...");
            System.Console.ReadKey();

        }
    }
}
```



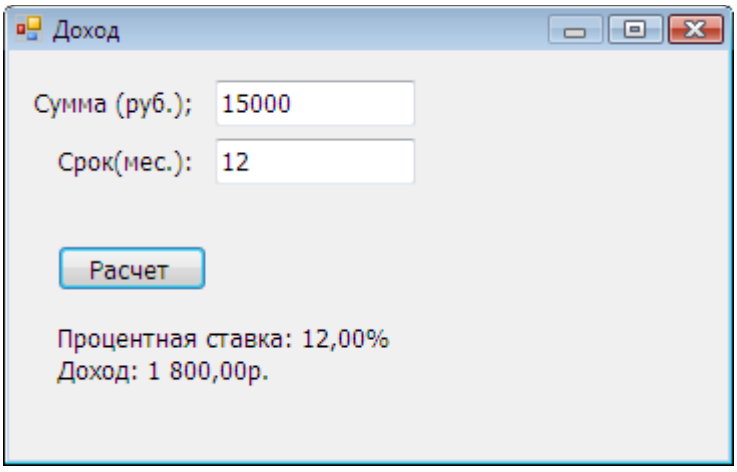


Windows Forms Application

Приложение

- Консольное
- Оконное (классическое)
- Windows Forms (Windows 7, Windows 8.1, Windows 10)
- WPF
- Мобильное (Windows Phone 7, Windows Phone 8)

UWP - Universal Windows Platform (Windows 10 на десктопе, смартфоне, планшете, ...)

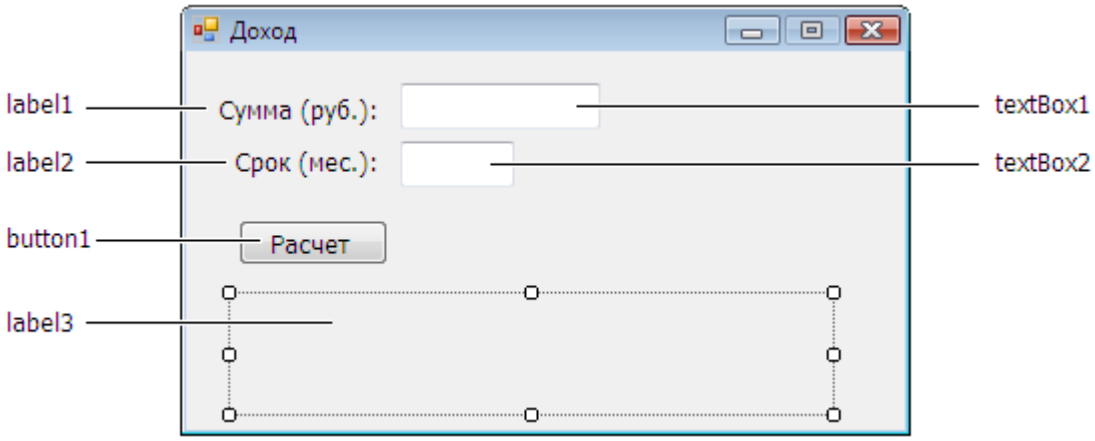


Windows Forms Application

Windows Forms Application:

- Постановка задачи
- Разработка
  - Разработка формы
  - Создание функций обработки событий
- Отладка
- Тестирование
- Публикация

Форма



Property (свойство) – определяет вид и поведение объекта.
Text, Size, Location, Enabled, Checked, Autosize,….

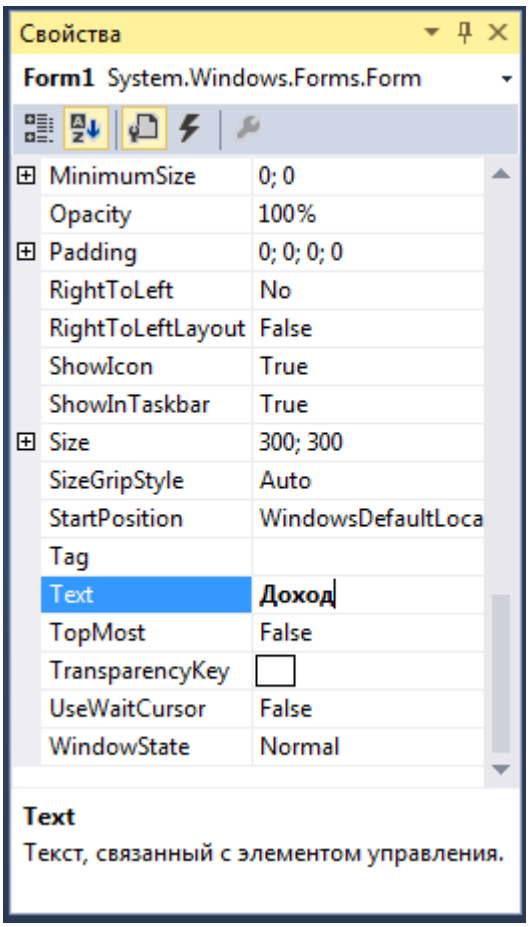
Компоненты

- Label
- TextBox
- Button
- RadioButton
- ListBox

Свойства

Property (свойство) – определяет вид и поведение объекта.

- Text
- SizeLocation
- Enabled
- Checked
- Autosize
- …

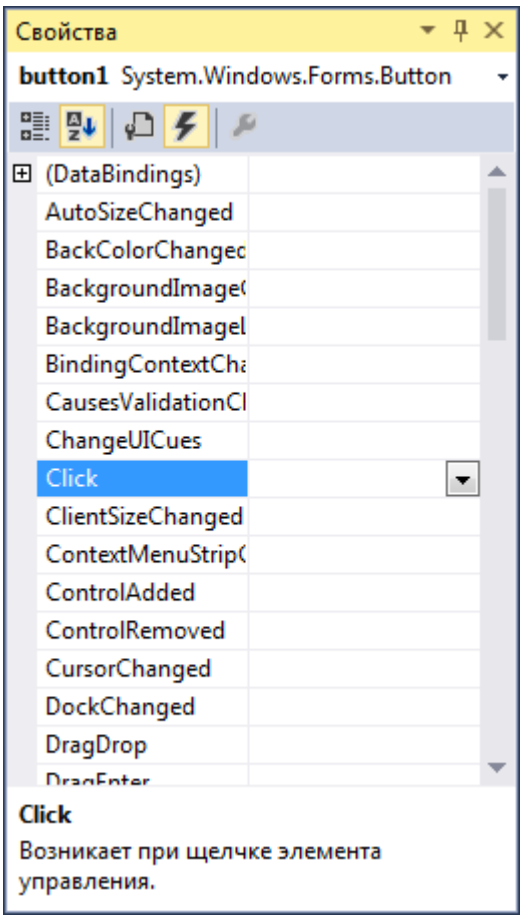


Событие

- Event (событие):
- Click
- KeyPress
- TextChange
- …

Создание функции обработки события

На вкладке Events найти событие и сделать двойной щелчок в поле рядом с именем события.



Функция обработки события

```
private void button1_Click(object sender, EventArgs e)
{

}
```

```
private void button1_Click(object sender, EventArgs e)
{
    double sum;       // сумма
    int    period;    // срок

     double percent;    // процентная ставка
     double profit;     // доход

     sum =  Convert.ToDouble(textBox1.Text);
     period =  Convert.ToInt32(textBox2.Text);

     if (sum < 10000)
           percent = 8.5;
     else
           percent = 12;

     profit = sum * (percent/100/12) * period;
```

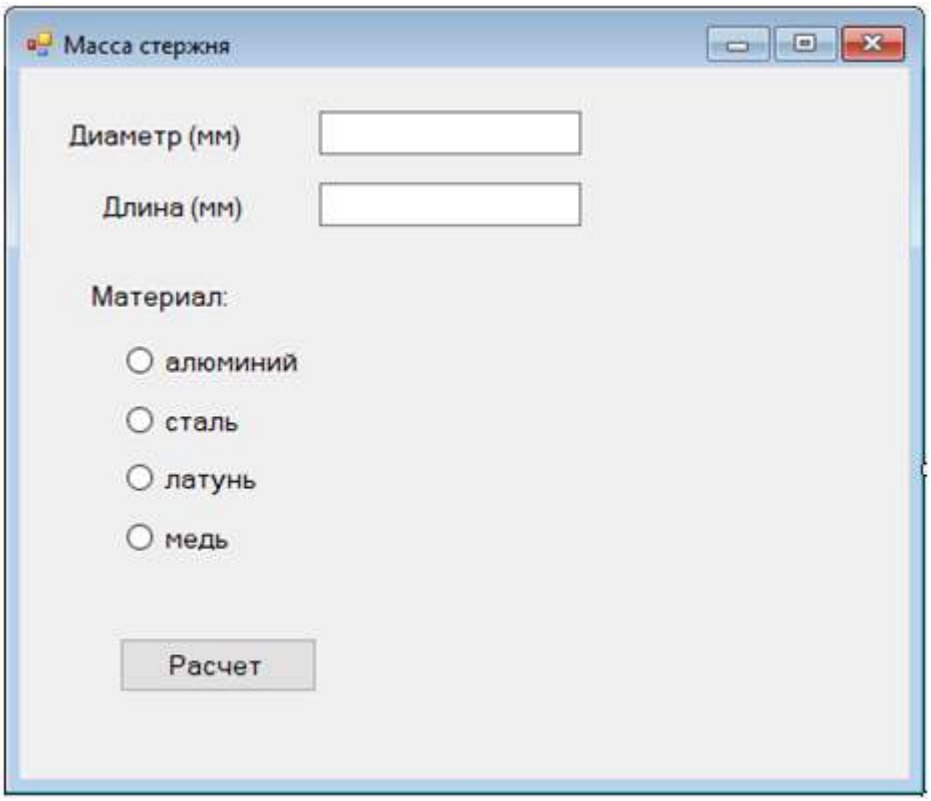
```
      label3.Text =
          String.Format("Процентная ставка:{0} "%\nДоход:  {1}“, percent, profit);
}

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
   if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
     return;

   // "Правильный" десятичный разделитель — запятая
   if (e.KeyChar == '.') e.KeyChar = ',';

   if (e.KeyChar == ',')  {
    // Нажата запятая. Запятая уже есть в поле редактирования?
    if ((textBox1.Text.IndexOf(',') != -1) ||  (textBox1.Text.Length == 0)) {
     // Запятая уже есть.
     e.Handled = true;
    }
    return;
   }

   if (Char.IsControl(e.KeyChar))  {
    if (e.KeyChar == (char)Keys.Enter)  {
     // Переместить курсор в поле Срок
      textBox2.Focus();
    }
    return;
   }

   // остальные символы запрещены
   e.Handled = true;
}

private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
{
   // в поле Срок можно ввести только целое число

  if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))
                    return;

   if (Char.IsControl(e.KeyChar))
   {
    if (e.KeyChar == (char)Keys.Enter) {
         // Переместить фокус на кнопку Расчет
      button1.Focus();
    }
    return;
   }
```

```
    // остальные символы запрещены
    e.Handled = true;
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    label3.Text = ""; // очистить поле отображения результата расчета

    if ((textBox1.Text.Length == 0) || (textBox2.Text.Length == 0))
        // В поле редактирования нет данных.
        // Сделать кнопку Расчет недоступной
        button1.Enabled = false;
    else
        // Сделать кнопку Расчет доступной
        button1.Enabled = true;
}

public Form1()
{
    InitializeComponent();

    // сделать кнопку Расчет недоступной
    button1.Enabled = false;
}
```

Компоненты

- RadioButton
- CheckBox
- ListBox

RadioButton



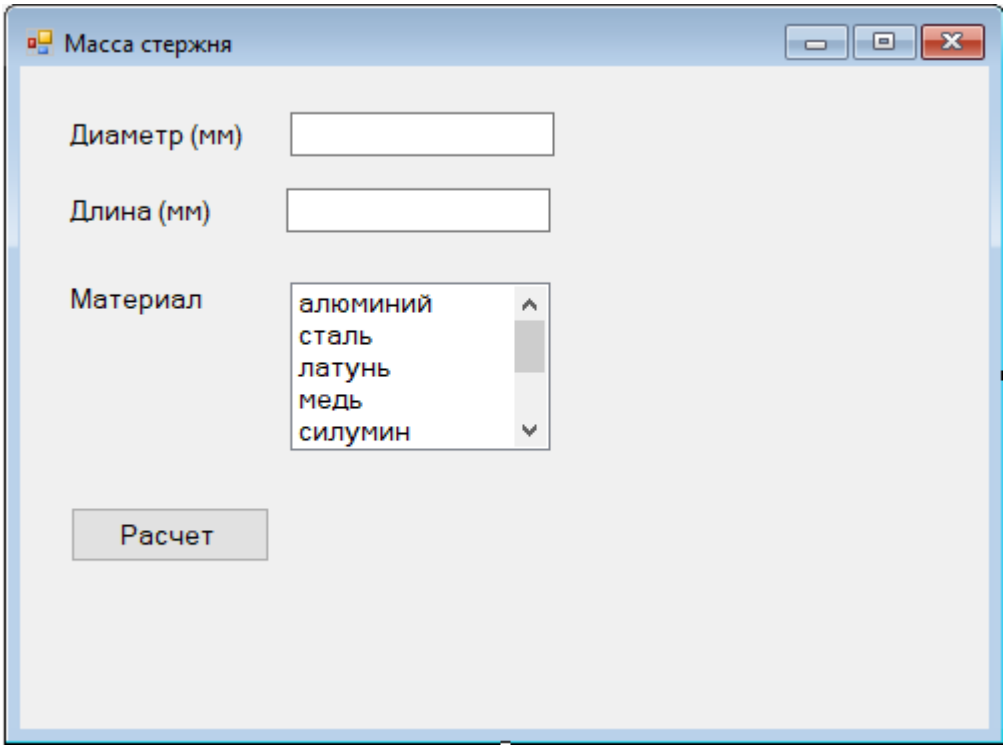
Property:

- Text
- Checked

Events:

- Click
- CheckChanged

ListBox



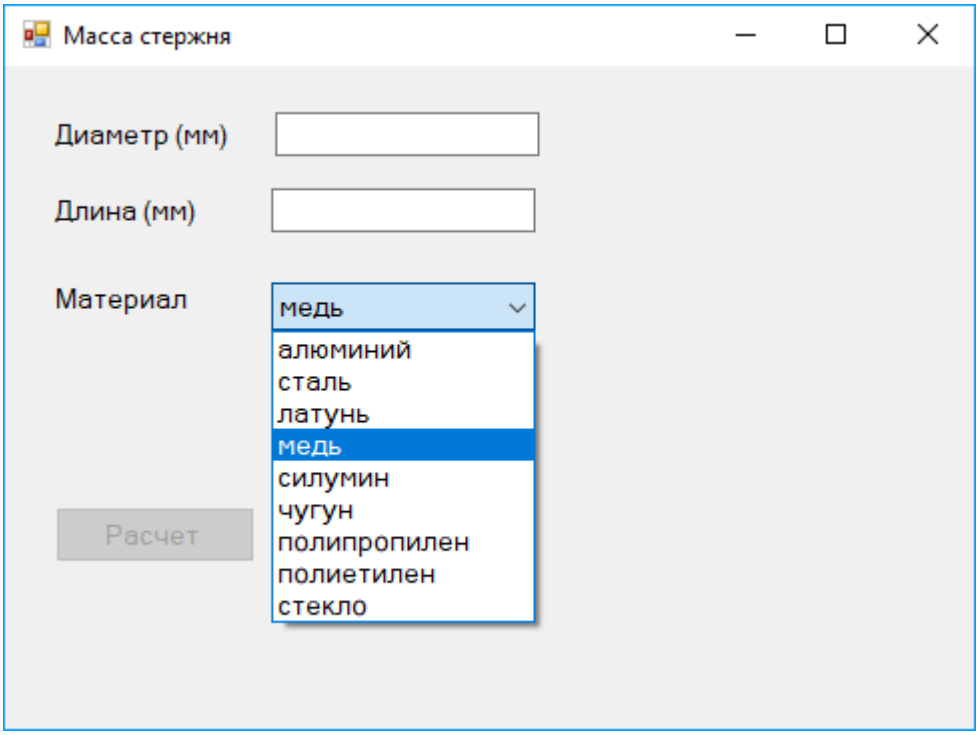
Property:

- Items
- SelectedItem
- SelectedIndex

Events:

- SelectedIndexChanged

ComboBox



Property:

- DropDpwnStyle = DropDown | DropDownList
- Items
- SelectedItem
- SelectedIndex

Events:

- SelectedIndexChanged

Инициализация списка коде

```
public partial class Form1 : Form
{
    string[] materaialTitle = {"полистирол","полиэтилен"};

    public Form1()
    {
      InitializeComponent();

      comboBox1.Items.Add("алюминий");
      comboBox1.Items.Add("сталь");
      comboBox1.Items.Add("латунь");

      for (int i=0; i < materaialTitle.Length; i++)
      {
```

```
            comboBox1.Items.Add(materaialTitle[i]);
        }
    }

}
```

Деловая графика

PictureBox

- Отображение иллюстраций
- Поверхность для формирования графики

Отображение иллюстраций

Property

- Image
- SizeMode

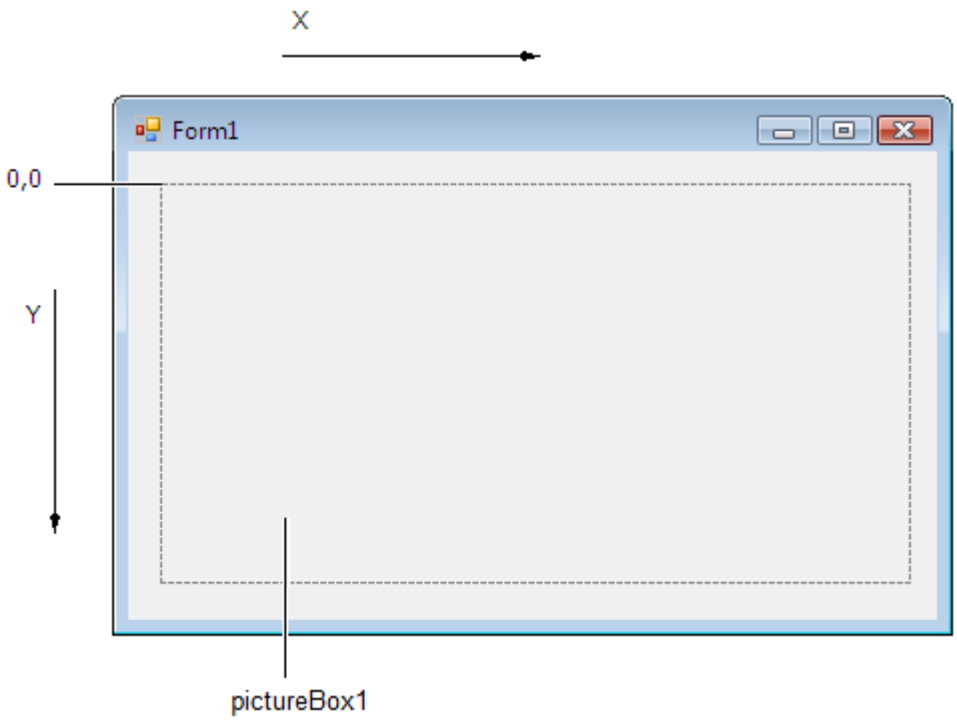
Формирование графики

Графика – совокупность графических примитивов

Компонент - PictureBox

Событие – Paint

Graphics – графическая поверхность



private void pictureBox1_Paint(object sender, PaintEventArgs e)

{

// e.Gaphics – графическая поверхность, на которой
// программа может рисовать

// методы объекта Graphics рисуют

/ / свойства объекта Graphics определяют вид графических примитивов

}

```
private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
    int x, y, w, h;
    int x0, y0;

    x0 = 10; y0 = 10;
    w = 28; h = 60;

    x = x0; y = y0;

    e.Graphics.FillRectangle(System.Drawing.Brushes.Green, x, y, w, h);

    x += w;
    e.Graphics.FillRectangle(System.Drawing.Brushes.White, x, y, w, h);

    x += w;
    e.Graphics.FillRectangle(System.Drawing.Brushes.Red, x, y, w, h);

    e.Graphics.DrawRectangle(System.Drawing.Pens.Black, x0, y0, w * 3, h);

    e.Graphics.DrawString("Италия", this.Font,
                    System.Drawing.Brushes.Black, x0, y0 + h + 10);

}
```

Доступ к Graphics есть только у функции обработки события Paint!

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void flag_it(Graphics graphics, int x0, int y0, int width, int height)
    {
        int x, y, w;

        x = x0; y = y0;
        w = width / 3;

        graphics.FillRectangle(System.Drawing.Brushes.Green, x, y, w, height);

        x += w;
        graphics.FillRectangle(System.Drawing.Brushes.White, x, y, w, height);

        x += w;
        graphics.FillRectangle(System.Drawing.Brushes.Red, x, y, w, height);

        graphics.DrawRectangle(System.Drawing.Pens.Black, x0, y0, w * 3, height);

        graphics.DrawString("Italia", this.Font, System.Drawing.Brushes.Black, x0, y0 + height + 10);
    }

    private void pictureBox1_Paint(object sender, PaintEventArgs e)
    {
        flag_it(e.Graphics, 10,10,84,60);
    }
}
```

Графические примитивы

Метод	**Действие**
DrawLine(Pen, x1, y1, x2, y2) DrawLine(Pen, p1, p2)	Рисует линию. Параметр Pen определяет цвет, толщину и стиль линии; параметры x1, y1, x2, y2 или p1 и p2 — координаты точек начала и конца линии
DrawRectangle(Pen, x, y, w, h)	Рисует контур прямоугольника. Параметр Pen определяет цвет, толщину и стиль границы прямоугольника: параметры x, y — координаты левого верхнего угла; параметры w и h задают размер прямоугольника
FillRectangle(Brush, x,y, w, h)	Рисует закрашенный прямоугольник. Параметр Brush определяет цвет и стиль закраски прямоугольника; параметры x, y — координаты левого верхнего угла; параметры w и h задают размер прямоугольника
DrawEllipse(Pen, x, y, w, h)	Рисует эллипс (контур). Параметр Pen определяет цвет, толщину и стиль линии эллипса; параметры x, y, w, h — координаты левого верхнего угла и размер прямоугольника, внутри которого вычерчивается эллипс

FillEllipse(Brush, x, y, w, h)	Рисует закрашенный эллипс. Параметр Brush определяет цвет и стиль закраски внутренней области эллипса; параметры x, y, w, h — координаты левого верхнего угла и размер прямоугольника, внутри которого вычерчивается эллипс
DrawPolygon(Pen, P)	Рисует контур многоугольника. Параметр Pen определяет цвет, толщину и стиль линии границы многоугольника; параметр P (массив типа Point) — координаты углов многоугольника
FillPolygon(Brush, P)	Рисует закрашенный многоугольник. Параметр Brush определяет цвет и стиль закраски внутренней области многоугольника; параметр P (массив типа Point) — координаты углов многоугольника
DrawString(str,Font,Brush,x, y)	Выводит на графическую поверхность строку текста. Параметр Font определяет шрифт; Brush — цвет символов; x и y — точку, от которой будет выведен текст
DrawImage(Image, x, y)	Выводит на графическую поверхность иллюстрацию. Параметр Image определяет иллюстрацию; x и y — координату левого верхнего угла области вывода иллюстрации

Относительные координаты и масштабирование

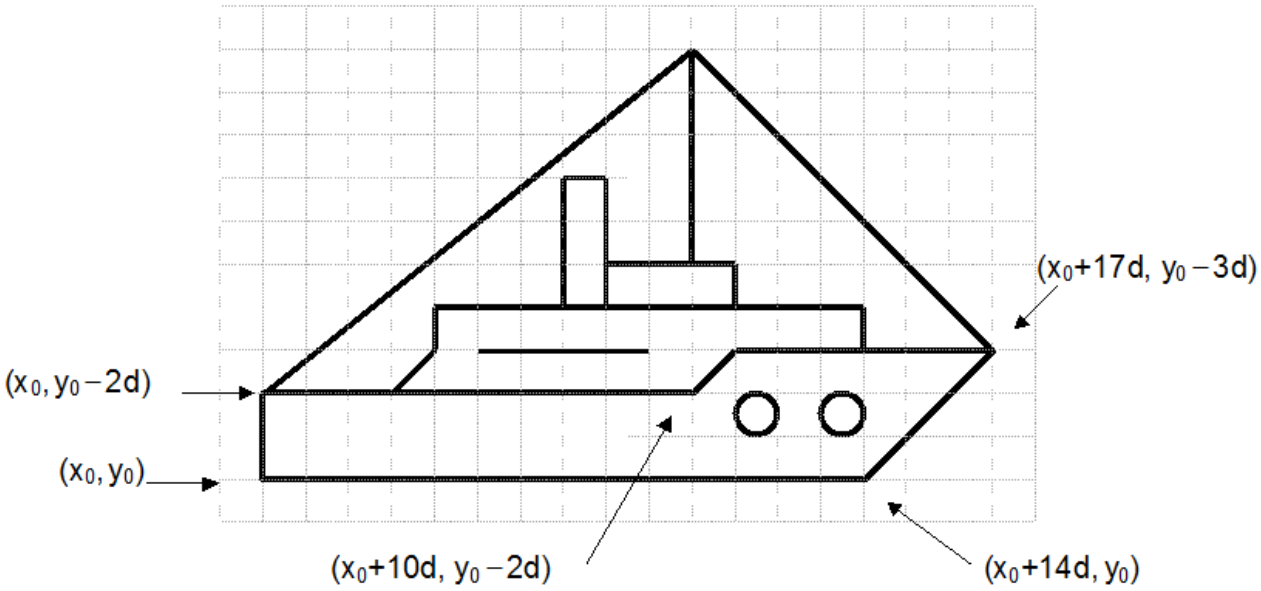


График (диаграмма)

- Данные загружаются из текстового файла в список (массив)
- Загрузка и обработка данных должна выполняться ОДИН РАЗ

- График должен занимать ВСЮ область компонента PaintBox

Структура программы

```
public partial class Form1 : Form
{

    // здесь данные, загруженные из файла

    public Form1()
    {
        InitializeComponent();
    }


     private void Form1_Activated(object sender, EventArgs e)
    {
        // загрузка данных из файла
    }


    private void drawGraph(Graphics graphics)
    {
        // показать границу компонента
        graphics.DrawRectangle(Pens.Black, 0, 0, pictureBox1.Size.Width-1, pictureBox1.Size.Height-1);

        // рисуем график

    }

    private void pictureBox1_Paint(object sender, PaintEventArgs e)
    {
        drawGraph(e.Graphics);
    }


}
```

Загрузка данных

```
public partial class Form1 : Form
{

    List<string> data;

    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Activated(object sender, EventArgs e)
    {
        data = new List<string>();

        String path = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

        string textFile = path + Path.DirectorySeparatorChar + "report.csv";

        if (System.IO.File.Exists(textFile))
        {
            System.IO.StreamReader sr = new StreamReader(textFile);
            while (!sr.EndOfStream)
            {
                string st = sr.ReadLine();

                data.Add(st);
            }
            sr.Close();
        }
        else
        {
           label1.Text =  "Нет файла данных:" + textFile;
        }
    }
}
```

Построение графика

```
private void drawDiagram(Graphics gr)
{
                // заголовок

                // обработка данных

                // построение графика

}
```

Заголовок

```
// шрифт подписей данных
Font dFont = new Font("Tahoma", 9);

// шрифт заголовка
Font hFont = new Font("Tahoma", 14, FontStyle.Regular);
string header = "Курс доллара";

// ширина области отображения текста
int w = (int)gr.MeasureString(header, hFont).Width;

int x = (pictureBox1.Width - w) / 2;

gr.DrawString(header, hFont, System.Drawing.Brushes.ForestGreen, x, 5);
```

Область построения графика

График строим в отклонениях от минимального значения ряда данных, так, чтобы он занимал всю область построения.

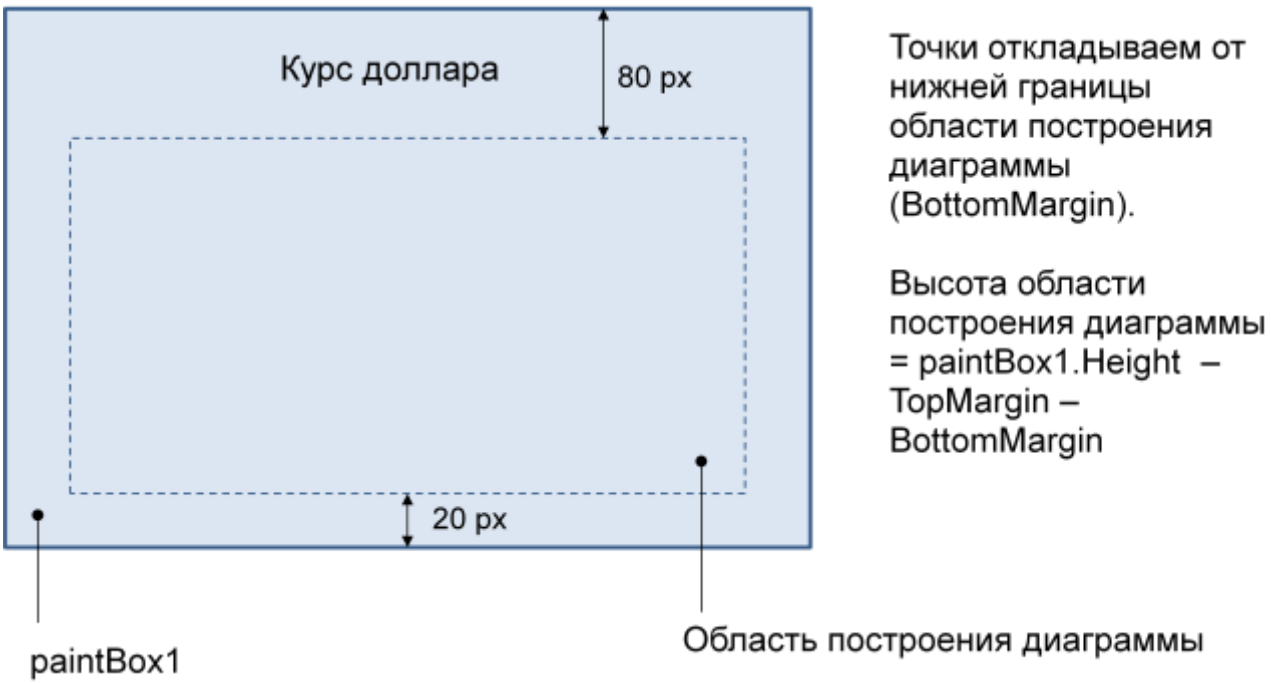
Область построения

TopMargin (отступ сверху) = 80;
BottomMargin(отступ снизу) = 20;
LeftMargin(отступ слева) = 20;
RightMargin(отступ справа) = 20.

Шаг по X

```
double[] d; // данные


// расстояние между точками графика (шаг по X)
int dx = (int)((pictureBox1.Width – (LeftMargin + RightMargin) / (d.Length - 1));
```

Координата Y



Точки откладываем от нижней границы области построения диаграммы (BottomMargin).

Высота области построения диаграммы = paintBox1.Height – TopMargin – BottomMargin

Yi = PictureBox.Height - BottomMargin – (d[i] – min) * K

Для d[max] :
TopMargin = PictureBox.Height - BottomMargin – (max - min)*K

K = (PictureBox.Height – BottomMargin – TopMargin) / (max – min)

```
// коэф. масштабирования
double K = (pictureBox1.Height - BottomMargin - TopMargin) / (max - min);



double max = d[0]; // максимальный элемент массива

double min = d[0]; // если строим в отклонениях

double min = 0;    // если все значения положительные и хотим
                   // отобразить значения, а не отклонения

// поиск min и max элементов
for (int i = 1; i < d.Length; i++)
{
    // ????
}



 // первая точка
 int x1 = 20;
 int y1 = pictureBox1.Height - BottomMargin - (int)((d[0] - min) * K);
```

```
 // маркер первой точки
 g.DrawRectangle(System.Drawing.Pens.Black, x1 - 2, y1 - 2, 4, 4);

 // подпись численного значения первой точки
 g.DrawString(Convert.ToString(d[0]), dFont,
            System.Drawing.Brushes.Black, x1 - 10, y1 - 20);


  // остальные точки
 for (int i = 1; i < d.Length; i++)
 {
    x2 = x1 + dx;
    y2 = pictureBox1.Height - BottomMargin - (int)((d[i] - min) *K);

    // маркер точки
    g.DrawRectangle(System.Drawing.Pens.Black, x2 - 2, y2 - 2, 4, 4);

    // соединим текущую точку с предыдущей
    g.DrawLine(System.Drawing.Pens.Black, x1, y1, x2, y2);

    // подпись численного значения
    g.DrawString(Convert.ToString(d[i]), dFont,
                System.Drawing.Brushes.Black, x2 - 10, y2 - 20);

    x1 = x2;
    y1 = y2;
 }
```

Базы данных

База данных и СУБД

- Microsoft SQL Server
- Oracle
- MySQL

- Microsoft Access

- SQLite

База данных на физическом уровне:

- Файл (совокупность файлов)

База данных на уровне логики:

- Совокупность записей

- Совокупность связанных таблиц – реляционная база данных (relation – связь, зависимость)

DB Projects

projects

pid	title	start	finish	managerID
1	Приложение Мои расходы	01.04.2018		4
2	База данных Поликлиника	01.05.2018		3

tasks

pojectID	title	start	finish	workerID
1	Программирование и отладка	15.04.2018		2
1	Разработка интерфейса	10.04.2018		1
1	Разработка структуры БД	01.04.2018		5
2	Программирование	10.05.2018		2
2	Разработка структуры БД	05.05.2018		1
2	Тестирование	20.05.2018		2

workers

workerID	Name	salary
1	Homer Simpson	6000
2	Bart Simpson	6500
3	Lisa Simpson	7000
4	Marge Simpson	7000
5	Maggie Simpson	6500

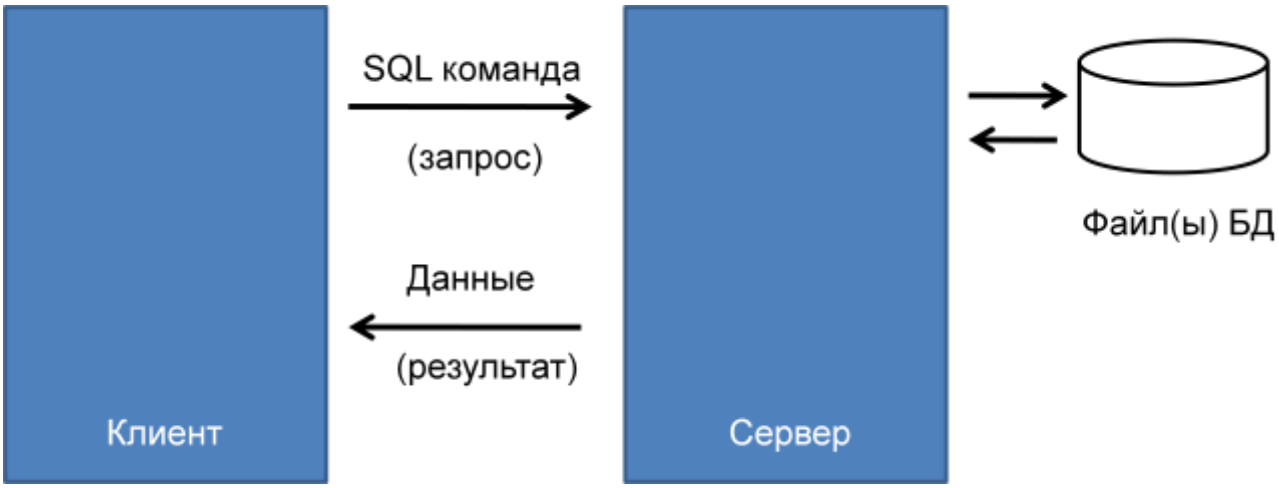
DB Books

books

Title	Author
Программирование на C#	Культин Н.Б.
Microsoft Visual C# в задачах и примерах	Культин Н.Б.
Microsoft Visual C++ в задачах и примерах	Культин Н.Б.
Delphi в задачах и примерах	Никита Культин
Основы программирования в Delphi	Культин Никита
Управление инновационными проектами	И.Л.Туккель, А.В. Сурина, Н.Б. Культин
Инструменты управления проектами: Project Expert и Microsoft Project	Н.Б.Культин
Язык программирования C++	Страуструп Б.
Программирование на C++	Дьюхарст С., Старк К.
Основы программирования в Microsoft Visual C++ 2010	Культин Н.Б.

Архитектура Клиент-сервер

client (англ.) – заказчик, пользователь покупатель
service (англ.) – обсуживать, обслуживание, услуга



Язык SQL

SQL - Structured Query Language (язык структурированных запросов)

- SELECT
- INSERT
- UPDATE
- DELETE

- CREATE TABLE

…

БД

- Локальная
- Удаленная

Microsoft Access

Локальная СУБД (система управления базами данных)
Использует ядро баз данных Microsoft Jet

.mdb
.mdbx

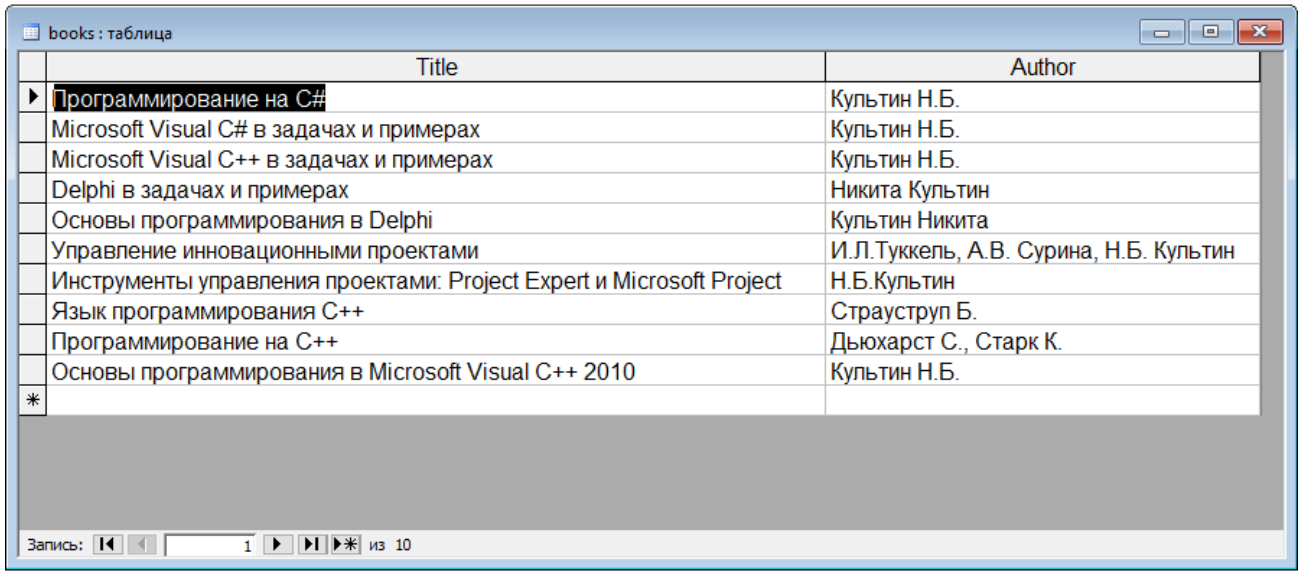
Создание БД в Microsoft Access

- Создать файл БД

- Открыть конструктор таблиц
- Ввести названия полей и задать их тип (строковый или числовой)
- Закрыть окно конструктора (задать имя таблицы)
- Открыть таблицу и ввести в нее информацию

Basic SQL commands

SELECT
INSERT
UPDATE
DELETE




```
SELECT Title, Author FROM books
WHERE Author Like 'Культин Н.Б.'
```

или

```
SELECT * FROM books
WHERE Author Like 'Культин Н.Б.'
```

```
SELECT * FROM books WHERE Author Like 'Культин Н.Б.'
```

```
SELECT * FROM books WHERE Author Like 'Культин'
```

```
SELECT * FROM books WHERE Author Like '%Культин%'
```
или
```
SELECT * FROM books WHERE Author Like '*культин*'
```

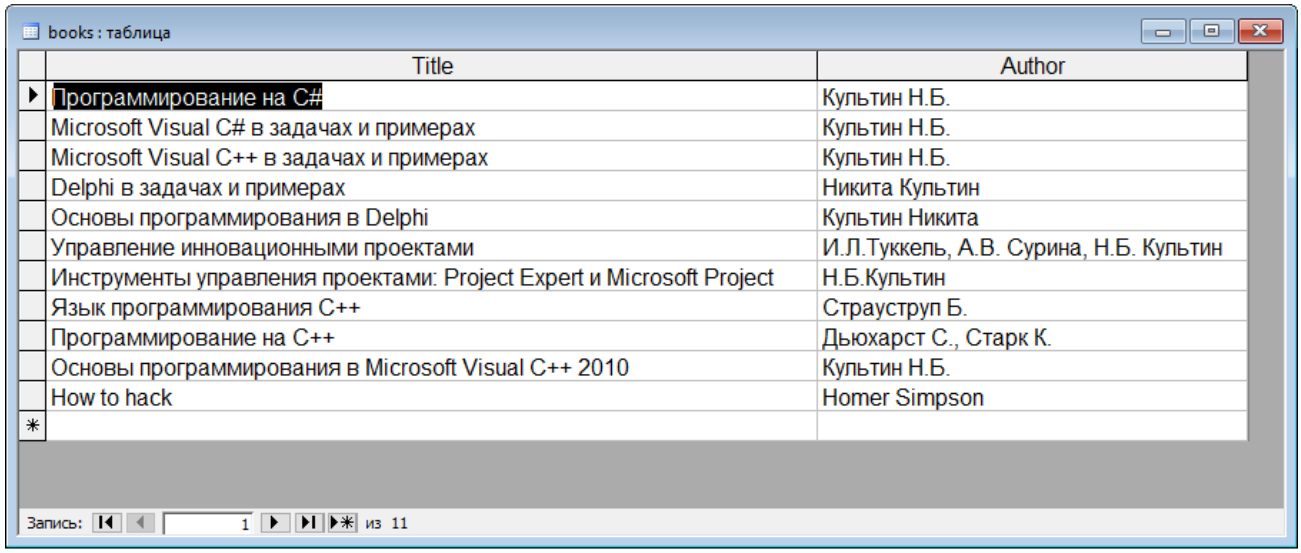
wildcards in Like

Access supports two dialects of SQL: Access SQL and ANSI-92 SQL. The latter is compatible with SQL Server; it uses % as wildcard for any number of characters, and _ for a single character, instead of * and ? respectively.
You can specify the version to use in File > Options > Object Designers > SQL Server Compatible Syntax (ANSI 92) in the Query design section.

```
SELECT * FROM books
WHERE (Title Like '*c++*') And (Author Like '*ку*')
```

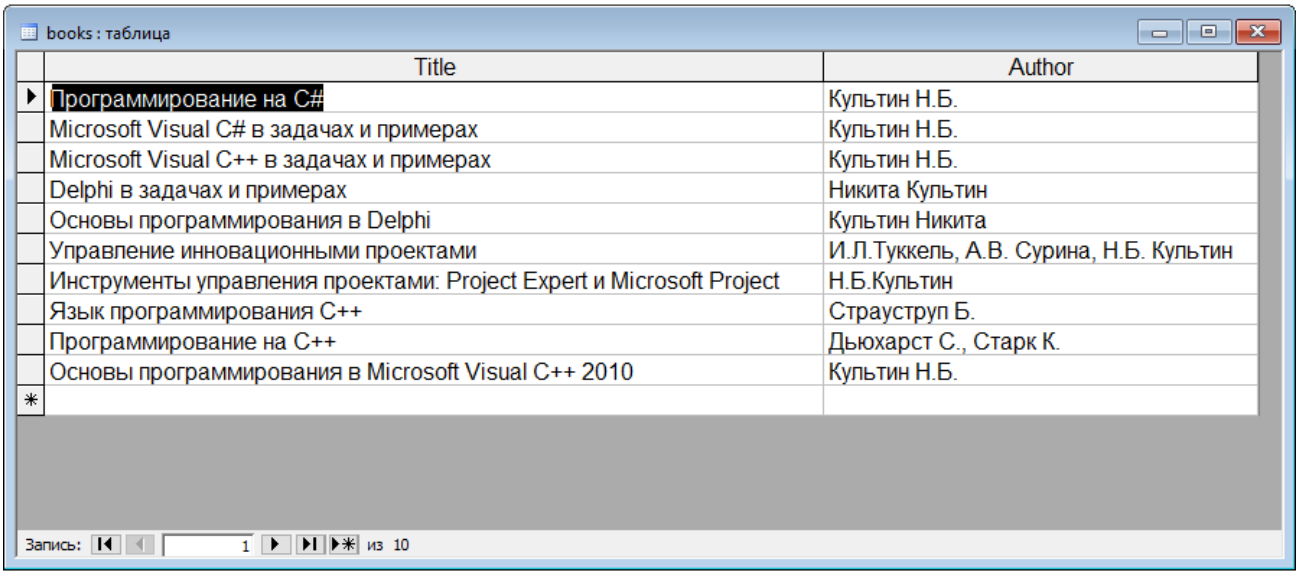
```
SELECT books.* FROM books
WHERE books.Title Like '*c++*'
ORDER BY books.Author
```

```
INSERT INTO books ( Author, Title )
VALUES ('Homer Simpson', 'How to hack')
```




```
UPDATE books SET Author = 'Bart Simpson'
WHERE Author='Homer Simpson'
```

```
DELETE * FROM books
WHERE (Author Like 'Bart Simpson')
```

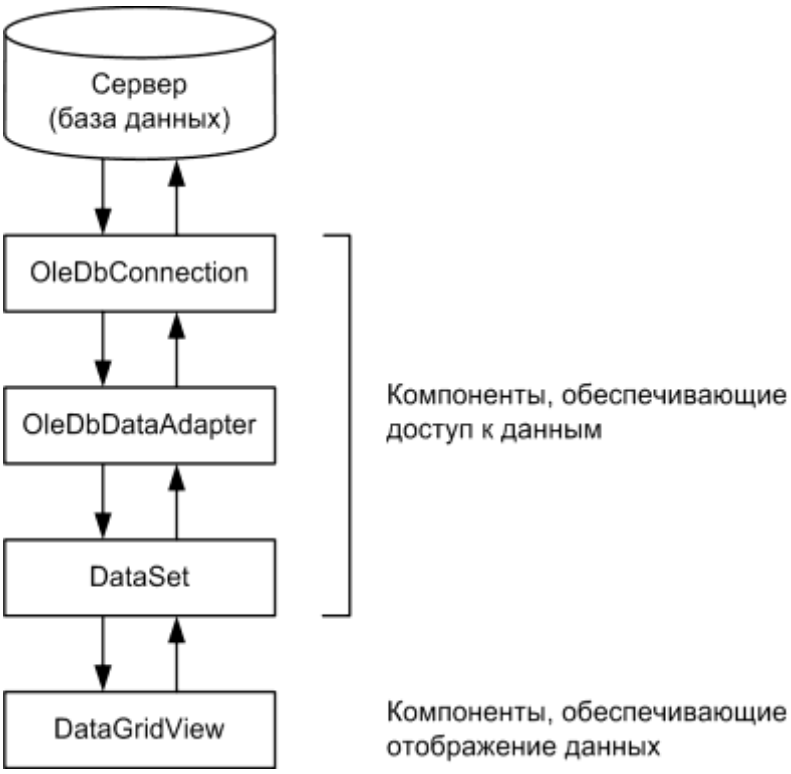


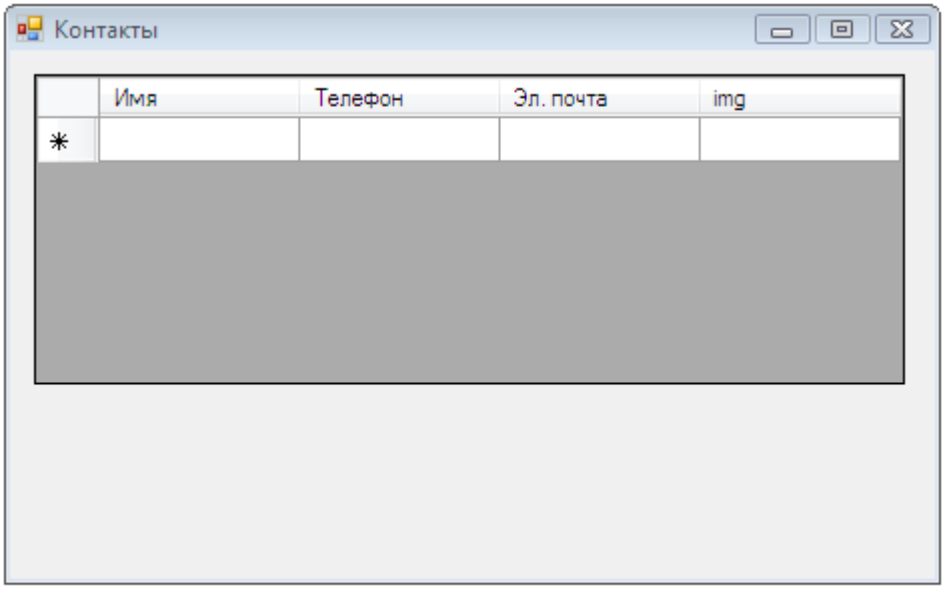
Технологии доступа к данным

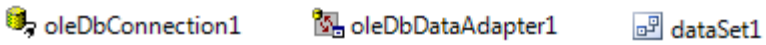
- ADO (от англ. ActiveX Data Objects — «объекты данных ActiveX») — интерфейс программирования приложений для доступа к данным
- ODBC (англ. Open Database Connectivity) — это программный интерфейс (API) доступа к базам данных
- **OLE DB (англ. Object Linking and Embedding, Database) — набор COM интерфейсов, которые позволяют приложениям унифицированно работать с данными разных источников и хранилищ информации.**
- Др.

Компоненты

- Доступ к базе данных обеспечивают компоненты OleDbConnection, OleDbDataAdapter и DataSet
- Отображение данных в табличной форме обеспечивает компонент DataGridView







Настройка компонентов

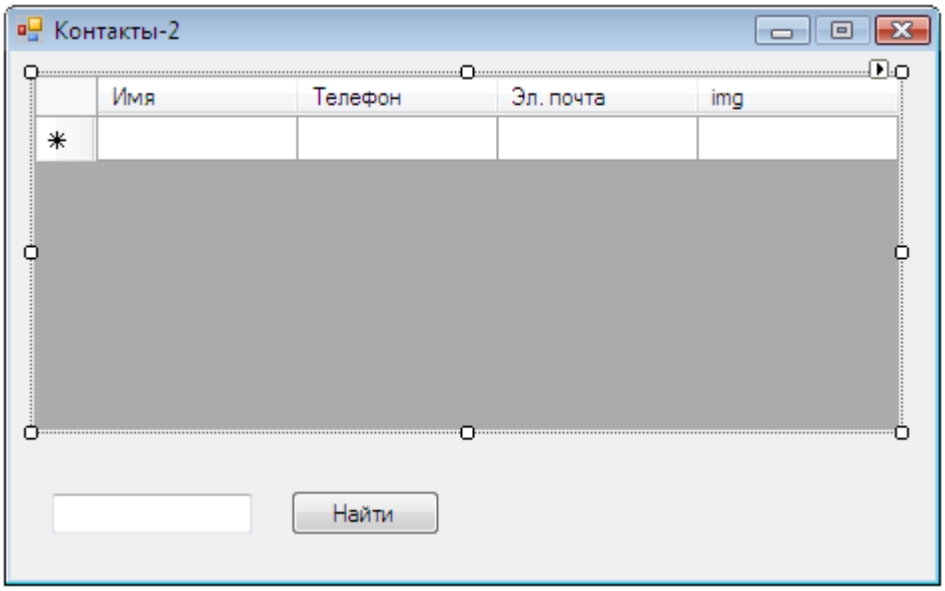
- OleDbConnection
- OleDbDataAdapter
- DataSet
- DataSet
- DataGridView

Обработка событий

```
 private void Form1_Load(object sender, EventArgs e)
 {
   // прочитать данные из БД
   oleDbDataAdapter1.Fill(dataSet1.Tables[0]);

   // можно и так:
   // oleDbDataAdapter1.Fill(dataSet1.Tables["contacts"]);
 }


// завершение работы программы
private void Form1_FormClosing(object sender,  FormClosingEventArgs e)
{
    oleDbDataAdapter1.Update(dataSet1.Tables[0]);
}

// пользователь выделил строку и нажал <Delete>
 private void dataGridView1_UserDeletingRow(object sender,
                        DataGridViewRowCancelEventArgs e)
{
    DialogResult dr = MessageBox.Show(
                "Запись будет удалена из БД.\nВыполнить?",
                "Удаление записи",
                MessageBoxButtons.OKCancel,
                MessageBoxIcon.Warning,
                MessageBoxDefaultButton.Button2);
     if (dr == DialogResult.Cancel)
     {
         e.Cancel = true;
     }
}
```

Параметры SQL команды

```
oleDbDataAdapter1.SelectCommand.CommandText
SELECT  * from contacts

oleDbDataAdapter1.InsertCommand.CommandText
INSERT INTO contacts (name, phone, email, img)
VALUES  (?, ?, ?, ?)

oleDbDataAdapter1.InsertCommand.Parameters
```



OleDataAdapter

Свойство	**Значение**
SelectCommand.Connection	oleDbConnection1
SelectCommand.CommandText	SELECT * FROM contacts WHERE name LIKE ?
SelectCommand.Parameters[0].ParameterName	name
SelectCommand.Parameters[0].SourceColumn	name
SelectCommand.Parameters[0].Value	%%

```
private void Form1_Load(object sender, EventArgs e)
{
   // получить информацию из БД
   // У команды SELECT есть параметр, который задает
   // критерий отбора записей, поэтому надо задать его значение
   oleDbDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
   oleDbDataAdapter1.Fill(dataSet1.Tables[0]);
}


// щелчок на кнопке Найти
private void button1_Click(object sender, EventArgs e)
{
    dataSet1.Clear(); // удалить старые данные

   // для получения информации из базы данных
   // используется команда SELECT с параметром:
   // SELECT *FROM contacts WHERE (name Like ? )
```

```
    // где: ? - параметр

    // задать значение параметра команды SELECT
    oleDbDataAdapter1.SelectCommand.Parameters["name"].Value =   "%" + textBox1.Text + "%";

    // выполнить команду
    oleDbDataAdapter1.Fill(dataSet1.Tables[0]);
}
```

Контрольные вопросы

- Перечислите этапы разработки программы
- Какой из этапов или какие этапы разработки программы являются, по вашему мнению, наиболее важными? Почему?
- Как убедиться, что программа правильная?
- В какой момент можно считать, что этап отладки завершен?
- Перечислите этапы жизненного цикла программы
- Для решения каких задач применяют ассемблер?
- Приведите примеры интерпретируемых языков программирования.
- Приведите примеры компилируемых языков программирования
- Когда обнаруживаются синтаксические ошибки в программе, разрабатываемой на интерпретируемом языке программирования?
- Когда обнаруживаются синтаксические ошибки в программе, разрабатываемой на компилируемом языке программирования?
- Нужна ли среда разработки на компьютере пользователя, чтобы он мог запустить программу, созданную на компилируемом языке программирования?
- Нужна ли среда разработки на компьютере пользователя, чтобы он мог запустить программу, созданную на интерпретируемом языке программирования?

- Перечислите способы представления алгоритмов
- Назовите основные элементы блок-схем алгоритмов
- Сформулируйте правило расстановки стрелок на блок-схеме алгоритма
- Перечислите алгоритмические структуры
- Закончите фразу: Условие это – выражение ...
- Закончите фразу: простое условие состоит из опе... и опе...
- Закончите фразу: сложное условие состоит из опе... и ...
- Перечислите логические операторы
- Сформулируйте принципы структурного программирования
- Перечислите правила хорошего стиля программирования

- Перечислите основные типы данных
- Назовите типы, которые можно использовать для представления вещественных значений
- Какое количество значащих цифр может иметь double значение?
- Назовите кодировки символов
- В каких кодировках символ занимает 1 байт памяти?
- Сколько байт может занимать символ в кодировке UTF8?
- В каких кодировках коды латинских букв одинаковые?
- В каких кодировках коды русских букв одинаковые?

- Какой номер у первого элемента массива?
- Какой метод следует использовать для поиска в упорядоченном массиве?
- Какой метод следует использовать для поиска в неупорядоченном массиве?

- Вещественный (double) массив A занимает в памяти компьютера 64 байта. Из скольких элементов состоит массив A? В каком диапазоне может меняться индекс элементов массива A?
- В чем разница между статическим и динамическим массивом (когда выделяется память для статического и динамического массивов)?

- Может ли базовый класс иметь доступ к полям, свойствам (методам) производного класса?
- Что такое «полиморфизм»?
- Какая функция называется виртуальной?
- В каком случае массив может состоять из объектов разного типа?
- Перечислите методы отладки программ.
- Ошибка записи формулы (не инструкции) это – какая ошибка?
- Какие ошибки устраняются в процессе отладки программы?
- Что такое “условная компиляция”?
- Перечислите режимы компиляции в Visual Studio.
- Надо ли в программе при помощи директивы препроцессору определять имя DEBUG, чтобы иметь возможность управления отладочной печатью.
- Что такое “исключение”?
- Перечислите типы исключений (приведите примеры)
- Приведите пример ситуации, в которой возможно исключение IndexOutOfRangeException.
- Что такое “тест”?

- Какой компонент следует использовать в качестве поверхности для отображения графики?
- Функция обработки какого события должна формировать графику?
- Сколько раз во время работы программы может возникать событие формы Activate?
- Сколько раз во время работы программы может возникать событие Paint?
- Какой объект обеспечивает отображение графики (свойства и методы какого объекта обеспечивают вывод графики)?
- Что такое графический примитив? Приведите примеры.
- Какой объект определяет цвет линии или границы прямоугольника?
- Какой объект определяет цвет закраски области, например, прямоугольника?
- Функция обработки какого события и какого объекта должна выполнять загрузку и обработку данных, которые надо представить в виде диаграммы или графика?
- Перечислите компоненты, обеспечивающие доступ к БД.
- Какой компонент обеспечивает отображение данных, полученных из таблицы БД?
- Таблица books базы данных Library состоит из столбцов Author, Title и Description. Запишите SQL команду, которая выбирает из БД информацию о книгах по программированию на C#. Результат выборки должен быть упорядочен по содержимому поля Author.
- В свойство какого компонента надо записать SQL команду, чтобы получить информацию из БД?

Литература

1. Фокс Дж. Программное обеспечение и его разработка: пер. с англ. – М.:Мир, 1985. – 368 с., ил.
2. Дал У., Дейкстра Э., Хорн К. Структурное программирование. – М.: Мир, 1975. – 243 с., ил.
3. Зелковиц М., Шоу А., Гэннон Дж. Принципы разработки программного обеспечения: пер. с англ. – М.:Мир, 1982. – 368 с., ил.
4. Вирт Н. Алгоритмы + структуры данных = программы
5. Вирт Н. Алгоритмы и структуры данных. М.: Мир, 1989. – 360 с., ил.
6. Страуструп Б. Язык программирования C++. Пер. с англ. Киев: ДиаСофт, 1993.
7. Дьюхарст С., Старк К. Программирование на C++. Пер. с англ. Киев: ДиаСофт, 1993. – 272 с. ил.
8. Культин Н.Б. Основы программирования в Microsoft Visual C# 2010. СПб.: БХВ-Петербург, 2011.
9. Культин Н.Б. Основы программирования в Microsoft Visual C++ 2010. СПб.: БХВ-Петербург, 2012.
10. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – 3 изд. испр. – СПб.: БХВ-Петербург, 2015. 320 с., ил.
11. Культин Н.Б. C/C++ в задачах и примерах. – 3 изд. испр. – СПб.: БХВ-Петербург, 2018. 256 с., ил.