

Дисциплины:
«Основы автоматизированной обработки данных» («двойка»),
«Технологии программирования» («тройка»)

**КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ
В ЭЛЕКТРОЭНЕРГЕТИКЕ**

5 семестр

Технологии программирования

6 семестр

Прикладное программирование

6 семестр

Обработка информации
в электроэнергетике

7 семестр

Математическое моделирование
в электроэнергетике

**ПРИМЕНЕНИЕ
КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

Основы автоматизированной обработки данных

Автоматизация решений технических задач

Информационное обеспечение
в электроэнергетике

Алгоритмизация задач
электроэнергетики

Интегрированный экзамен (7 семестр)

Цель текущего семестра: введение в современные технологии *разработки* прикладного программного обеспечения с использованием объектно-ориентированного языка **C#** в *среде программирования MS Visual Studio*

- **Первый полусеместр :**

- лекции/практики (они же лекции) - 4 часа в неделю;

- **контрольная работа 1 (на восьмой неделе)**

- Темы КР: Числовые константы. Скалярные объекты встроенных типов данных. Выражения, операции и преобразование типов. Блочный оператор. Операторы циклов, Вложенные циклы. Одномерные и двумерные массивы;*

- **Второй полусеместр :**

- практики (они же лекции) - 2 часа в неделю;

- лабораторные работы (**№№1-4**)

- 4 часа в неделю в компьютерных классах

- (Э311/Э316);

- По каждой лабораторной работе оценивается:*

- *выполнение базовой части - до 5 баллов,*
 - *выполнение индивидуального задания, оформления и защита отчёта*
- **остальные баллы;**

- **контрольная работа 2 (на последней неделе)**

- Темы КР: Объявление и элементы класса. Конструкторы класса. Свойство. Статические элементы класса. Наследование. Скрытие наследуемых элементов;*

- **Сессия: зачёт** в зачётную неделю в письменной форме.

- Курсовая работа:**

- задание выдаётся после выполнения обязательной части **ЛР№3**,
 - выполняется и демонстрируется на стационарных компьютерах в компьютерном классе, оформляется отчёт

Технологическая карта

1. Лекции: коэффициент значимости совокупных результатов лекционных занятий – 0.3

Текущая аттестация на лекциях	Сроки(дата начала - дата окончания)	Максимальная оценка в баллах
контрольная работа 1	01/09/2019 - 26/10/2019	100

Весовой коэффициент значимости результатов текущей аттестации по лекциям – 0.5

Промежуточная аттестация по лекциям – зачет

Весовой коэффициент значимости результатов промежуточной аттестации по лекциям – 0.5

2. Практические занятия: коэффициент значимости совокупных результатов практических занятий – 0.2

Текущая аттестация на практических/семинарских занятиях	Сроки(дата начала - дата окончания)	Максимальная оценка в баллах
контрольная работа 2	28/10/2019 - 21/12/2019	100

Весовой коэффициент значимости результатов текущей аттестации по практическим/семинарским занятиям – 1.0

Промежуточная аттестация по практическим занятиям – (не предусмотрено)

Весовой коэффициент значимости результатов промежуточной аттестации по практическим/семинарским занятиям – 0.0

3. Лабораторные занятия: коэффициент значимости совокупных результатов лабораторных занятий – 0.5

Текущая аттестация на лабораторных занятиях	Сроки(дата начала - дата окончания)	Максимальная оценка в баллах
Выполнение лабораторных работ (индивидуальных заданий) №1=20, №2=30, №3=10, №4=40	28/10/2019 - 21/12/2019	100

Весовой коэффициент значимости результатов текущей аттестации по лабораторным занятиям – 1.0

Промежуточная аттестация по лабораторным занятиям – (не предусмотрено)

Весовой коэффициент значимости результатов промежуточной аттестации по лабораторным занятиям – 0.0

4. Курсовая работа: коэффициент значимости совокупных результатов курсовой работы

Весовой коэффициент текущей аттестации выполнения курсовой работы/проекта – 0.0 (не предусмотрено)

Весовой коэффициент промежуточной аттестации выполнения курсовой работы/проекта – защиты – 1.0

Рекомендуемая литература :

1. **Шилдт, Герберт** С# 4.0: полное руководство.: Пер. с англ.-М.: ООО «И.Д.Вильямс», 2013.-1056с.:ил.
2. **Ватсон Б.** С# 4.0 на примерах. - СПб.:БХВ-Петербург, 2011.-608с.: ил.
3. **Эндрю Троелсен.** Язык программирования С# 2008 и платформа .NET 3.5 Framework. 4-е изд.: Пер с англ. М.: ВИЛЬЯМС, 2009. 1168с. ил.
4. **Джейсон Прайс, Майк Гандерлой.** Visual С# /NET. Полное руководство: Пер. с англ. К.: ВЕК+, СПб.: КОРОНА принт, К.: НТИ, М.: Энтроп, 2008. 960 с.
5. **Павловская Т.А.** С#. Программирование на языке высокого уровня. Учебник для вузов. СПб.: Питер, 2009. 432с.: ил.
6. **С# на примерах** – Спб.:Наука, 2016 . – 304с., ил.
7. **Язык С#: краткое описание и введение в технологии программирования:** учебное пособие / О. М. Котов. - Екатеринбург: Изд-во Урал. ун-та, 2014. 208 с.

Введение

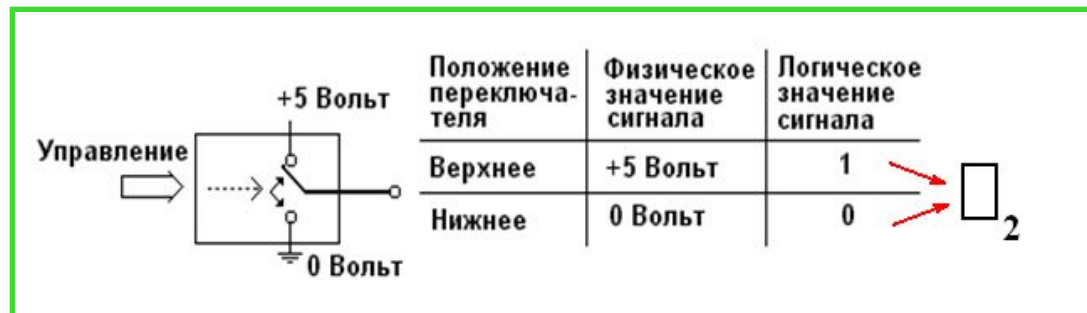
Не секрет, что **информационные технологии** – это наиболее динамично и масштабно развивающаяся сфера современного производства.

Основу любой цифровой информационной системы, как известно, составляют **электронные устройства**, выполняющие **программируемую обработку данных**. Среди этих устройств **компьютер** является основным и в наибольшей степени универсальным техническим средством оперирования с **информацией** (далее речь пойдёт только о компьютерах).

Информация существует в виде **сигналов**.

Сигнал – это технически различимый **параметр** некоторого **физического процесса**. В электронных устройствах такими процессами являются **электрические процессы**, а сигналом является уровень **напряжения**.

Цифровой сигнал характеризуется **конечным** набором **разрешённых уровней**. Количество уровней, в свою очередь, определяет используемую систему счисления (**позиционную**). На современном уровне развития электроники оптимальным являются **два уровня напряжения** и, соответственно, **двоичная (каноническая) система счисления**:



Системы счисления

Система счисления	Основание	Алфавит системы счисления
Двоичная	2	0,1
Десятичная	10	0,1,2,3,4,5,6,7,8,9
Шестнадцатеричная	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Формула значения числа

$$(a_{n-1}a_{n-2} \dots a_1a_0, a_{-1}a_{-2} \dots a_{-m})_b = \sum_i a_i b^i$$

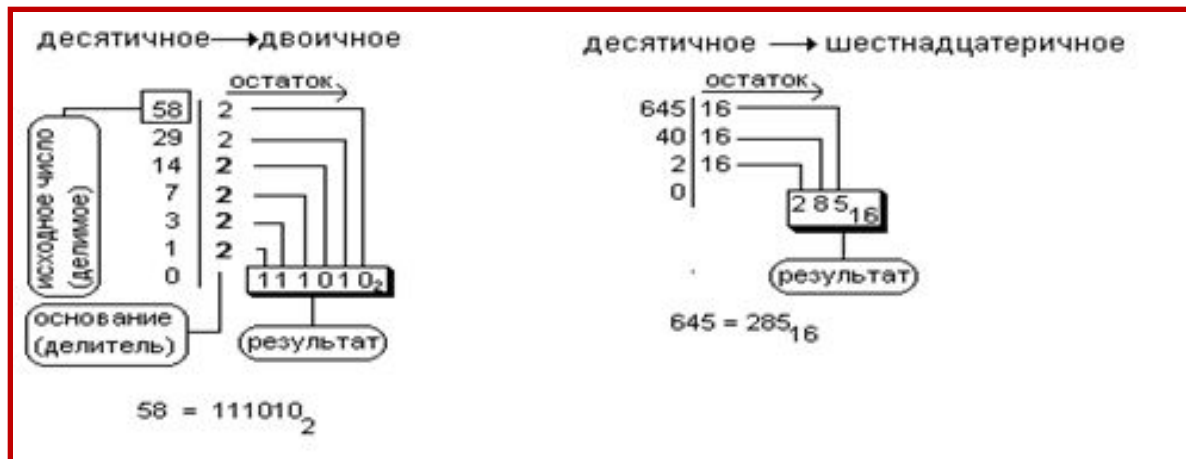
Целая часть

Дробная часть

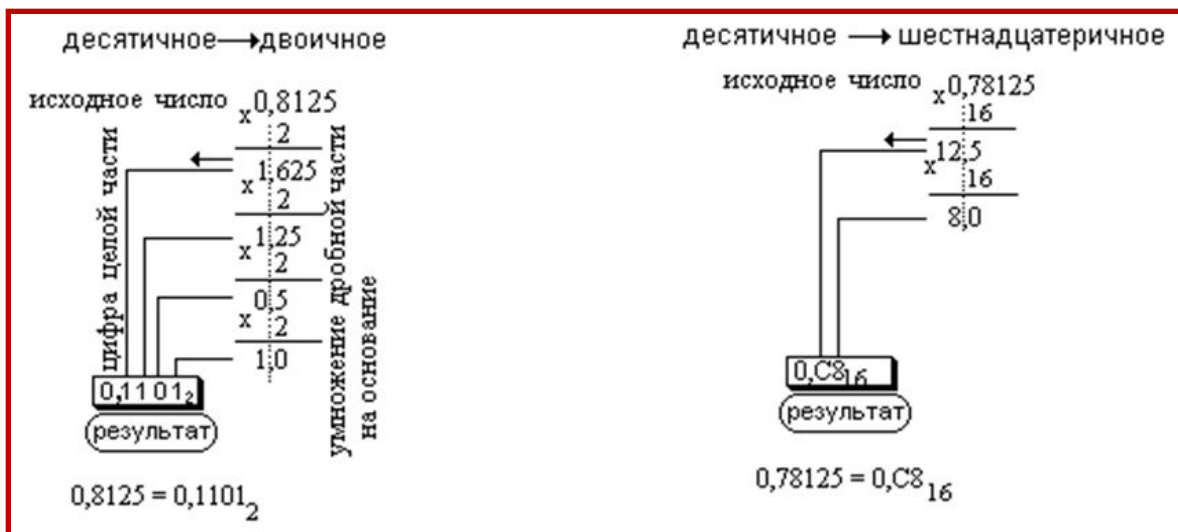
$$\begin{aligned} & \underbrace{(a_{n-1}a_{n-2} \dots a_1a_0)}_{\text{Целая часть}}, \underbrace{a_{-1}a_{-2} \dots a_{-m}}_{\text{Дробная часть}})_b = \\ & = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \dots + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} + \dots + a_{-m}b^{-m} \end{aligned}$$

Формула значения числа позволяет выполнить преобразование из любой **недесятичной системы** в **десятичную**

Последовательное *деление с остатком*:
 преобразование из *десятичной* системы в *недесятичную целой* части числа



Последовательное *умножение*:
 преобразование из *десятичной* системы в *недесятичную дробной* части числа



Преобразование между шестнадцатеричной и двоичной системами

основано на «*родственности*» их оснований: каждые *четыре разряда* двоичной системы соответствуют *одной цифре* шестнадцатеричной системы:

Система счисления		
<i>десятичная</i>	двоичная	шестнадцатеричная
<i>0</i>	0000	0
<i>1</i>	0001	1
<i>2</i>	0010	2
<i>3</i>	0011	3
<i>4</i>	0100	4
<i>5</i>	0101	5
<i>6</i>	0110	6
<i>7</i>	0111	7
<i>8</i>	1000	8
<i>9</i>	1001	9
<i>10</i>	1010	A
<i>11</i>	1011	B
<i>12</i>	1100	C
<i>13</i>	1101	D
<i>14</i>	1110	E
<i>15</i>	1111	F

Примеры преобразований и вычислений

$$1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

$$1234_{16} = 1 \cdot 16^3 + 2 \cdot 16^2 + 3 \cdot 16^1 + 4 \cdot 16^0 = 4660$$

$$00111010,11_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 32 + 16 + 8 + 2 + 0.5 + 0.25 = 58.75$$

$$A7_{16} + 1EF_{16} = 296_{16}$$

$$\begin{array}{r} A7_{16} \\ + 1EF_{16} \\ \hline 296_{16} \end{array}$$

n	7	6	5	4	3	2	1	0
2 ⁿ	128	64	32	16	8	4	2	1

$$000111010,001011_2 = 3A,2C_{16}$$

n	-1	-2	-3	-4
2 ⁿ	0,5	0,25	0,125	0,0625

Для самостоятельного решения

$$A24_{16} + 2CC_{16} = ?_{16}$$

$$110,1111_2 = ?_{16}$$

$$46 + 101001110_2 + 1C_{16} = ?$$

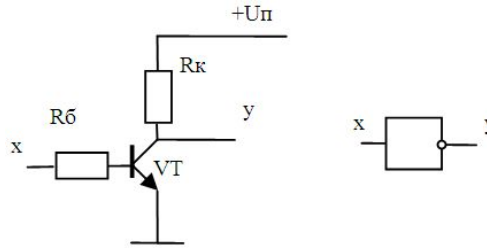
$$CA4_{16} + FF_{16} = ?_{16}$$

$$101,0111_2 = ?_{10}$$

$$1D_{16} + 110101110_2 = ?_{10}$$

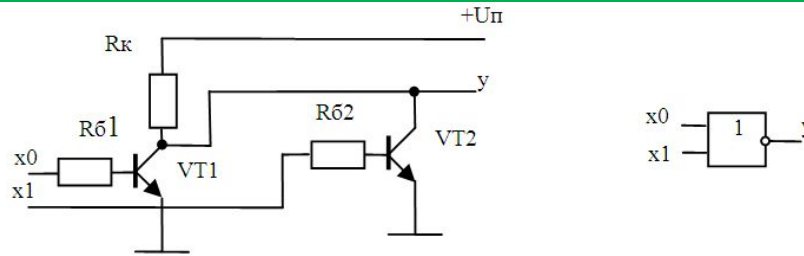
Элементы схемотехники

Инвертор:



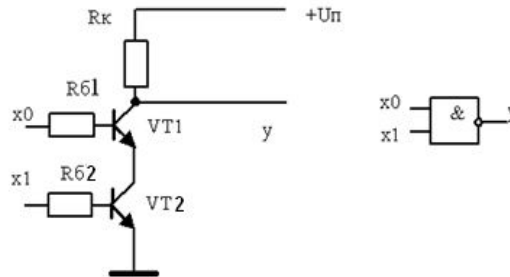
x	y
0	1
1	0

Элемент
2ИЛИ-НЕ:



x1	x0	y
0	0	1
0	1	0
1	0	0
1	1	0

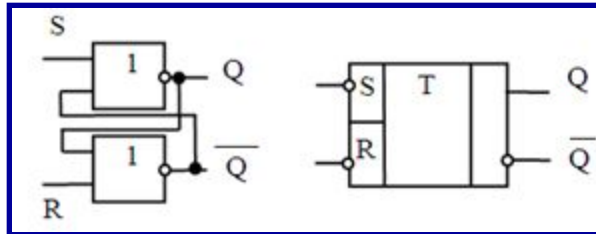
Элемент
2И-НЕ:



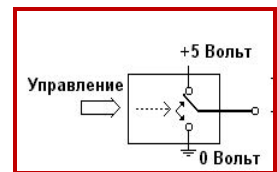
x1	x0	y
0	0	1
0	1	1
1	0	1
1	1	0



Триггер:
размещени
е двоичного
разряда



R	S	Q^{n+1}
0	0	Q^n
0	1	0
1	0	1
1	1	Неопределённое состояние



Внутреннее представление числовых данных

Беззнаковые целочисленные форматы

Байтовый (8 разрядов)

7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	1

Номер разряда

Содержимое разряда

Формула
максимального числа
в n разрядах:

$$\max = b^n - 1$$

диапазон [0, ..., 255] – 256 значений

Короткий целый (2 байта, 16 разрядов)

Старший байт								Младший байт							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	0	0	1	1	1	1	0	0	1

диапазон [0, ..., 65 535]

Средний (двойное слово, 32
разряда)

диапазон [0 ... 4 294 967 295]

Знаковые целочисленные форматы

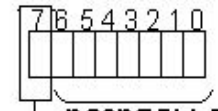
Байтовый

значение знакового разряда :

0 - число ≥ 0

1 - число < 0

↑ знаковый разряд



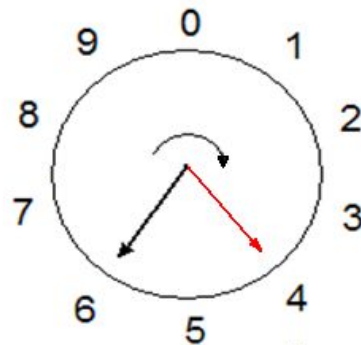
разряды для значения числа

Целый формат	Число разрядов	Диапазон значений	Ширина диапазона
Байтовый	8	-128 ... 127	256
Короткий	16	-32 768 ... 32 767	65536
Средний	32	-2 147 483 648... 2 147 483 647	4 294 967 296

Представление отрицательных чисел в форме *дополнений до двух*

Основным устройством процессора является *сумматор*, предназначенный прежде всего для *сложения* двоичных чисел. Для того, чтобы сумматор мог выполнять ещё и операцию *вычитания*, отрицательные числа должны быть представлены в форме

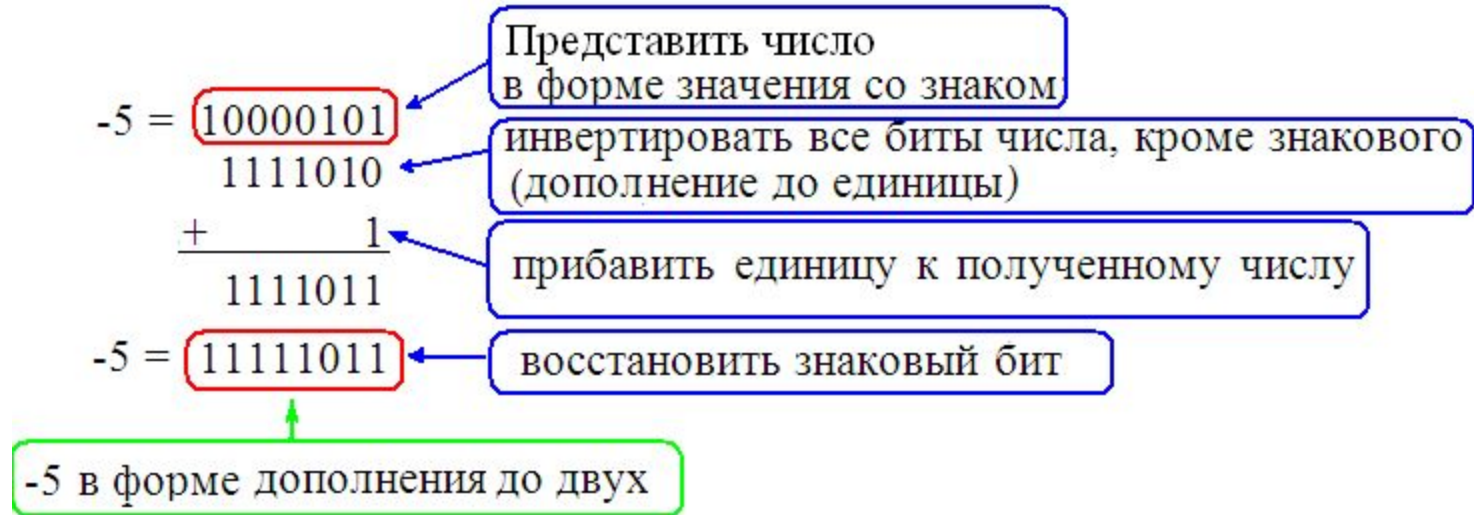
дополнений до двух



$$6 - 2 = 6 + (-2) = 6 + \text{Доп}_{10}(-2) = 6 + 8 = 4$$

$$\text{Доп}_{10}(-2) = 10 - 2 = 8$$

Порядок получения *дополнения до двух*



Пример

$$6_{10} - 15_{10} = ?$$

$$\begin{array}{r}
 -15 = 10001111_2 \\
 1110000_2 \\
 + \quad 1 \\
 \hline
 11110001_2
 \end{array}$$

дополнение до двух числа -15

$$\begin{array}{r}
 6 = 00000110_2 \\
 \longrightarrow 00000110_2 \\
 + 11110001_2 \\
 \hline
 11110111_2
 \end{array}$$

результат в ФДД,
т.к. отрицательный

$$\begin{array}{r}
 11110111_2 \\
 0001000_2 \\
 + \quad 1 \\
 \hline
 10001001_2 = -9
 \end{array}$$

окончательный
результат

Ситуация переполнения

Примеры переполнений

$65 + 65 = ?$

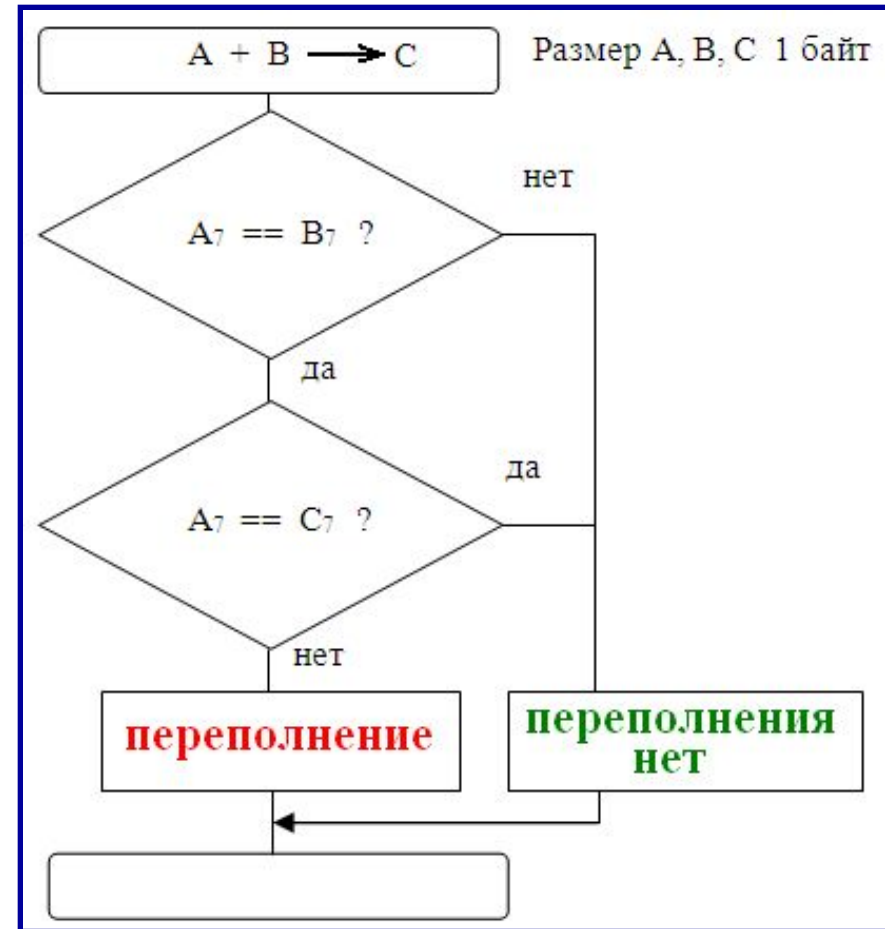
```
01000001
+01000001
-----
10000010
11111101
+      1
-----
11111110 = -126
результат
```

$-125 - 125 = ?$

Доп₂(-125) = 10000011

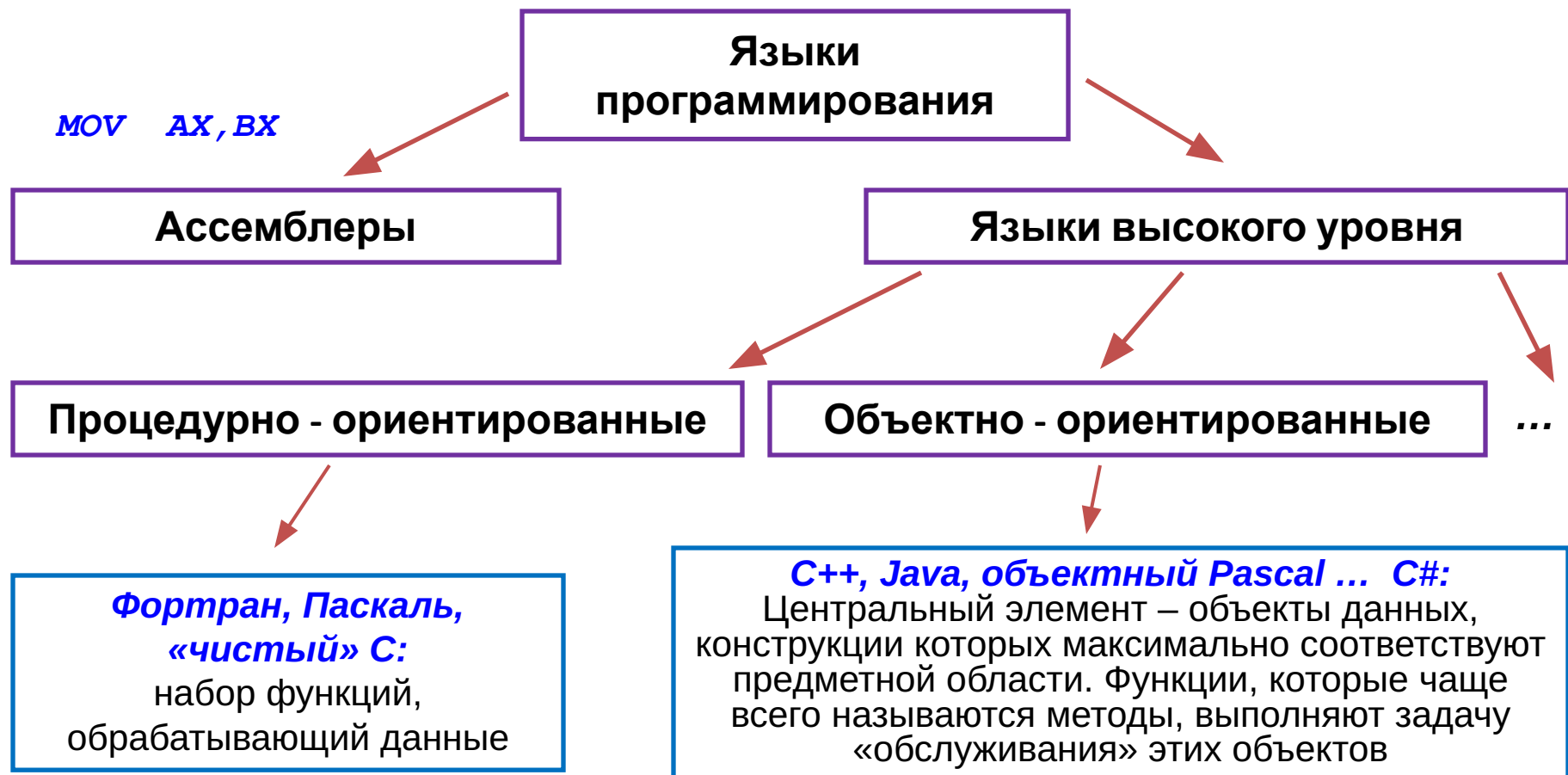
```
10000011
+10000011
-----
00000110 = 6
результат
```

Алгоритм проверки



Языки программирования

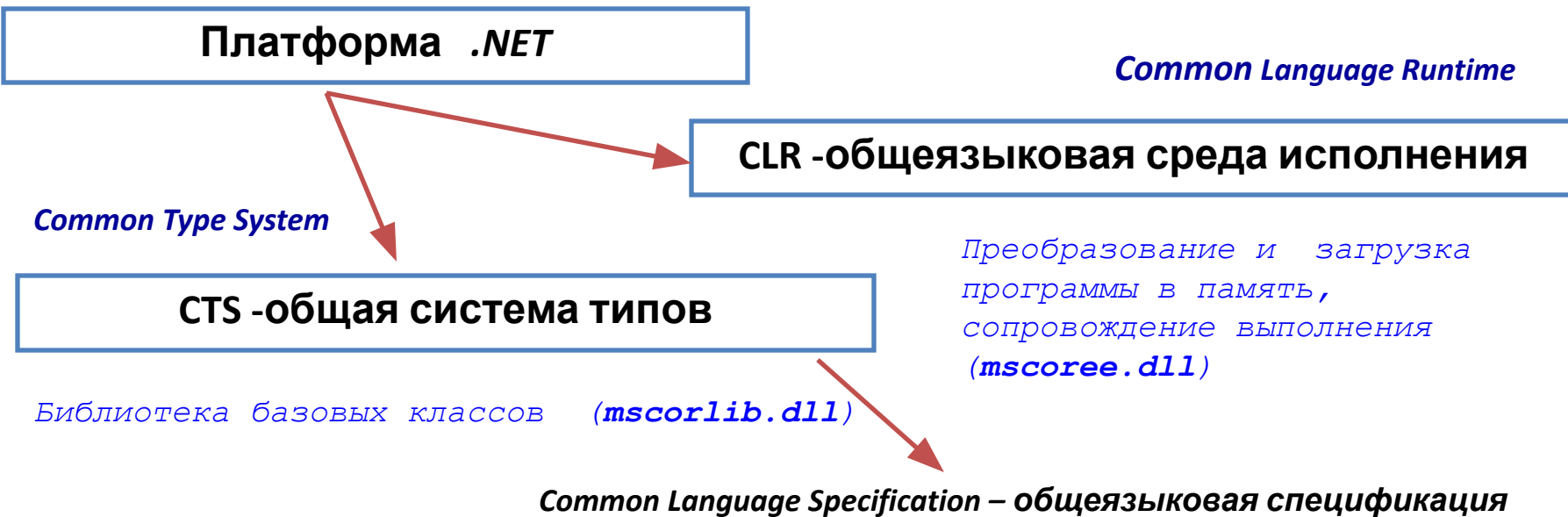
-это способ записи исходного кода программ для реализации того либо иного алгоритма обработки данных. Выполняет эту обработку, как известно, процессор.
По степени «близости» к процессору языки программирования делятся на *низкоуровневые* (машинно-ориентированные) и *высокоуровневые*.
Языки *высокого уровня*, прежде всего, различаются на *процедурно-* и *объектно-*ориентированные



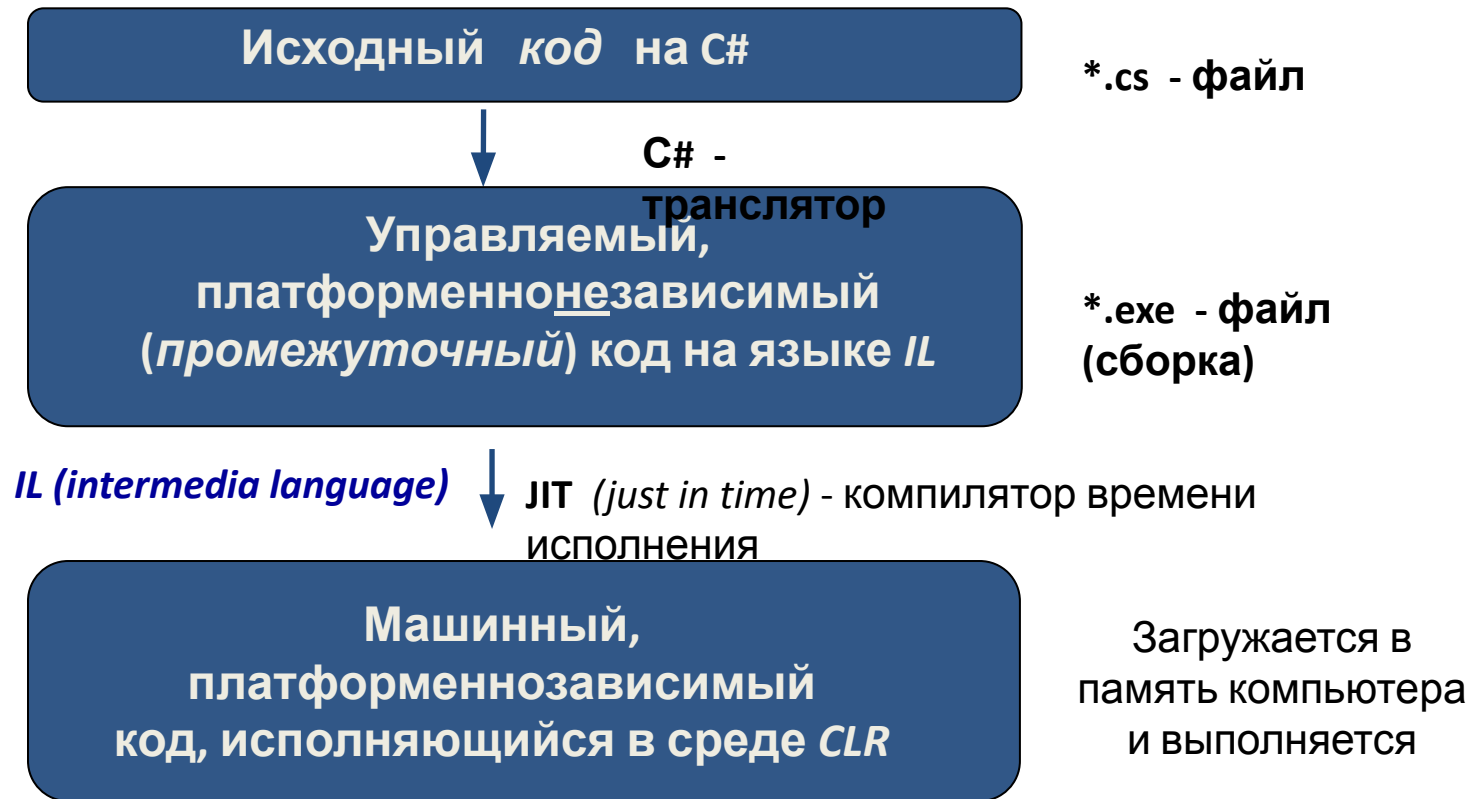
Первое представление об языке С#

Язык программирования С# (*произносится «си шарп», хотя название задумывалось, как «си дизель»*) вобрал в себя лучшие черты своих предшественников и в большей степени отвечает современным потребностям:

- Преемственность (в значительной степени) с языками С, С++, Java;
- Проще, чем С++, с сопоставимой «мощностью» и эффективностью;
- Полностью объектно-ориентированный (любая сущность - объект);
- Компонентно – ориентированный;
- Основан на платформе «.NET» (*произносится «дот нет»*);

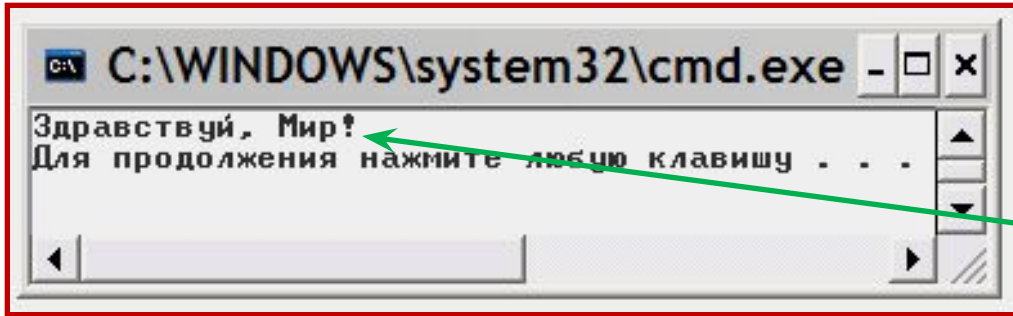


Технология создания приложения на C#



Наряду с языком C#, на платформе .NET могут быть использованы ещё несколько десятков других языков программирования (C++, VisualBasic, JScript и другие)

Первая программа



```
C:\WINDOWS\system32\cmd.exe - □ ×
Здравствуй, Мир!
Для продолжения нажмите любую клавишу . . .
```

С лёгкой руки разработчиков ещё языка C первая программа на новом (для разработчика) языке должна выводить на консольное окно (то есть на экран в текстовом режиме) простое приветствие

void Main() - заголовок метода `Main()`: объявляет пустой возвращаемый тип (`void`) и пустой список входных параметров.

WriteLine - вызов функции (метода) с входным аргументом - строковой константой

static объявляет метод `Main()` статическим, что определяет возможность его запуска на выполнение без явного создания объекта класса;

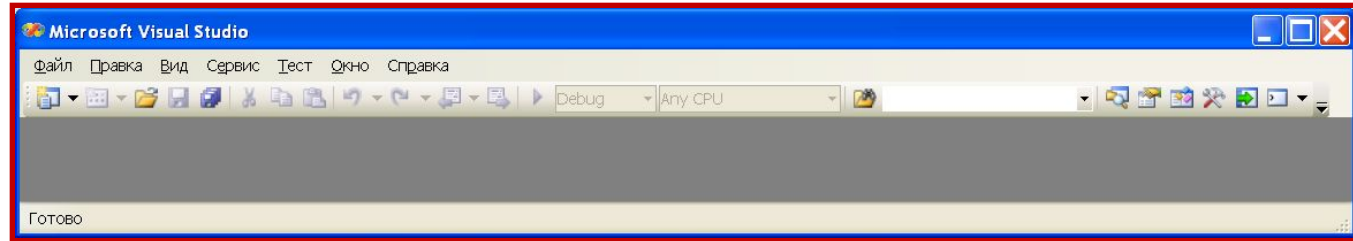
```
class FirstProgram
{
    static void Main()
    {
        System.Console.WriteLine("Здравствуй, Мир!");
    }
}
```

class - это ключевое слово для задания конструкции объекта, FirstProgram - имя этой конструкции, а метод `Main()` - часть этого объекта;

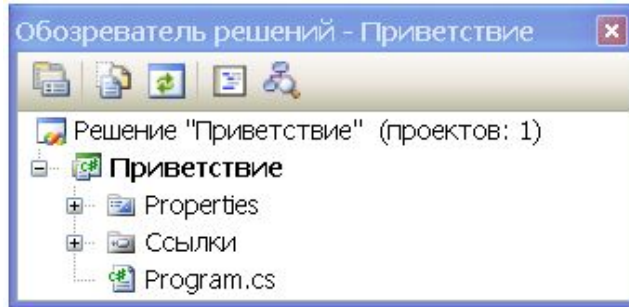
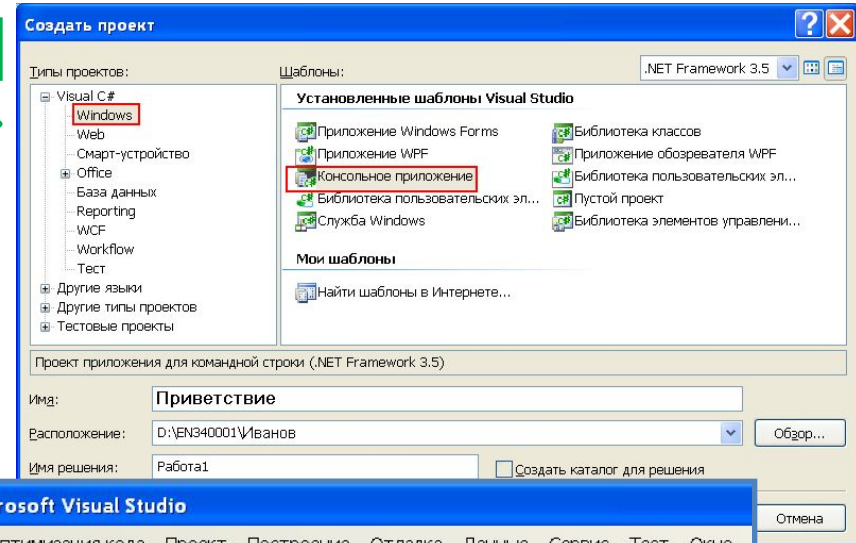
System.Console указывает, что метод `WriteLine()` является элементом класса Console в корневом пространстве имён System

Первая программа в среде Visual Studio (режим консольного приложения)

1. Запустить среду

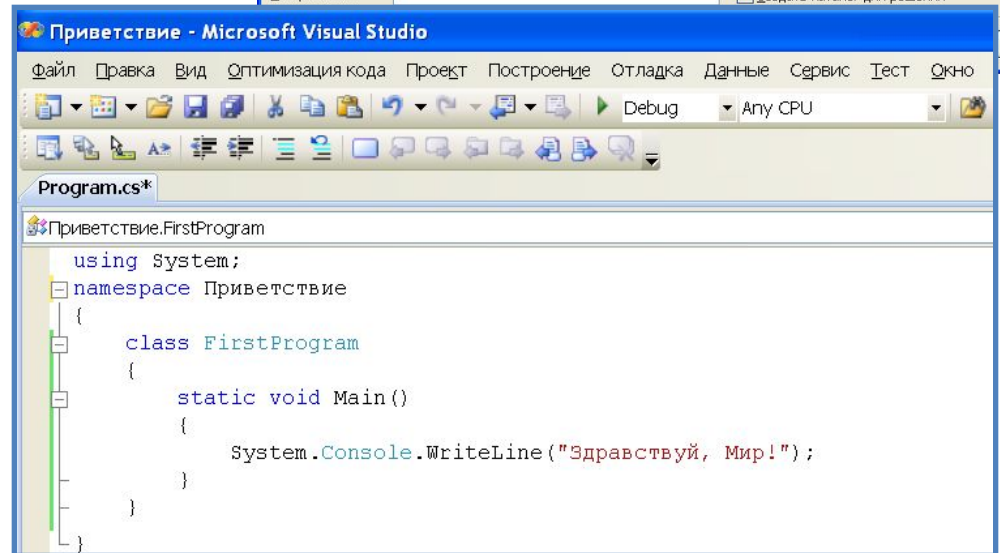


2. Создать проект **Файл -> Создать -> Проект**



3. В файле **Program.cs**
(откроется
автоматически)
рекомендуется
переименовать класс и
набрать одну строчку

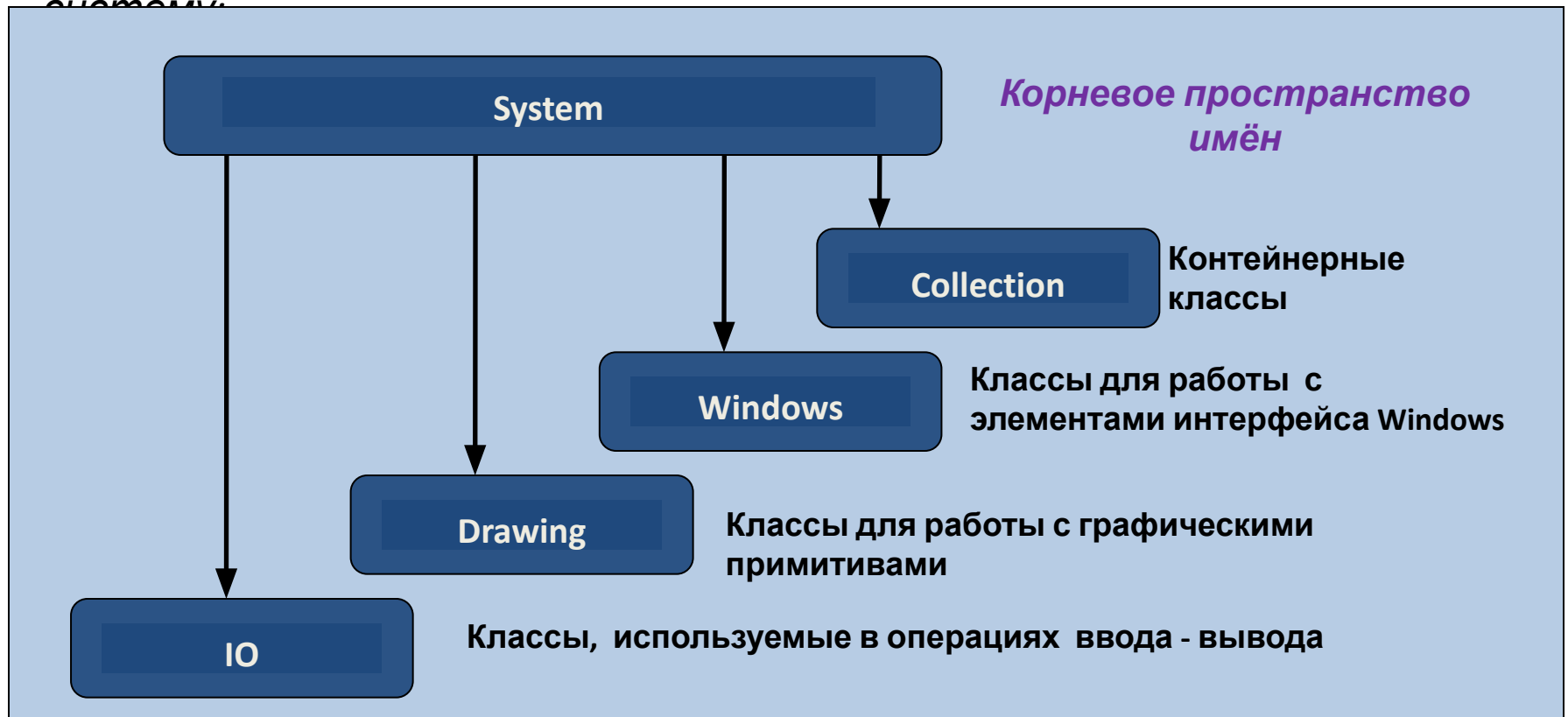
4. Запустить на выполнение
с помощью:
Отладка -> Запуск без отладки



Пространства имён

Система (библиотека) типов CTS насчитывает более **4000** различных типов, которые для удобства работы объединены в группы – пространства имён.

Группы, в свою очередь, представляют собой иерархическую древовидную систему.



Общая характеристика языка

Алфавит (набор литер)

языка программирования **C#** составляют символы:

- строчные и прописные **буквы** латинского алфавита;
- **цифры** от 0 до 9;
- символ **подчеркивания** ' _ ';
- набор специальных символов: " { } | [] + - % / \ ; ' : ? < > = ! & # ~ *
- прочие символы.

В свою очередь, алфавит **C#** служит для построения слов, которые называются **лексемами**. Различают пять типов **лексем**:

• **ключевые слова**;

• **разделители**;

• **идентификаторы**;

• **константы**;

• **знаки (символы) операций**;

Обособляют лексемы

Имена объектов, элементов

Неизменяемые данные

Основные, предназначены для записи текста **программ (кода)**

Компактная запись действий с данными

Ключевые слова

языка программирования **C#** составляют 77 конструкций, большая часть которых заимствованы из языков C, C++ и Java:

abstract
as
base
bool
break
byte
case
catch
char
checked
class
const
continue
decimal
default
delegate
do
double
else
enum
event
explicit
extern

false
finally
fixed
float
for
foreach
goto
if
implicit
in
int
interface
internal
is
lock
long
namespace
new
null
object

params
private
protected
public
readonly
ref
return
sbyte
sealed
short
sizeof
stackalloc
static
string
struct
switch
operator
out
override

this
throw
true
try
typeof
uint
ulong
unchecked
unsafe
ushort
using
virtual
void
volatile
while

Разделители

обособляют *лексемы* языка и могут быть следующих видов:

- пробел,
- табуляция,
- символ новой строки,
- скобки,
- комментарии

Комментарий – это текст, который предназначен только для читающего программу человека и игнорируется компилятором. В **C#** это можно сделать одним из двух способов :

□ **парный комментарий** -

/ коммен-
тарий */*

□ **строчный комментарий** –

// комментарий - заканчивается в конце данной строки

□ **xml – комментарий** (для создания самодокументируемых программ)

/// комментарий

Идентификаторы

- это *имена*, которые назначает программист. Именуются объекты, классы, структуры, методы, метки и т.п.

Правила написания идентификаторов достаточно простые:

- состоят из одного или более символов;
- идентификатор не может совпадать с ключевым словом;
- размер идентификатора не ограничен;
- первый символ – обязательно буква;
- можно использовать русские буквы;

Обычно даётся осмысленное, отражающее назначение элемента имя

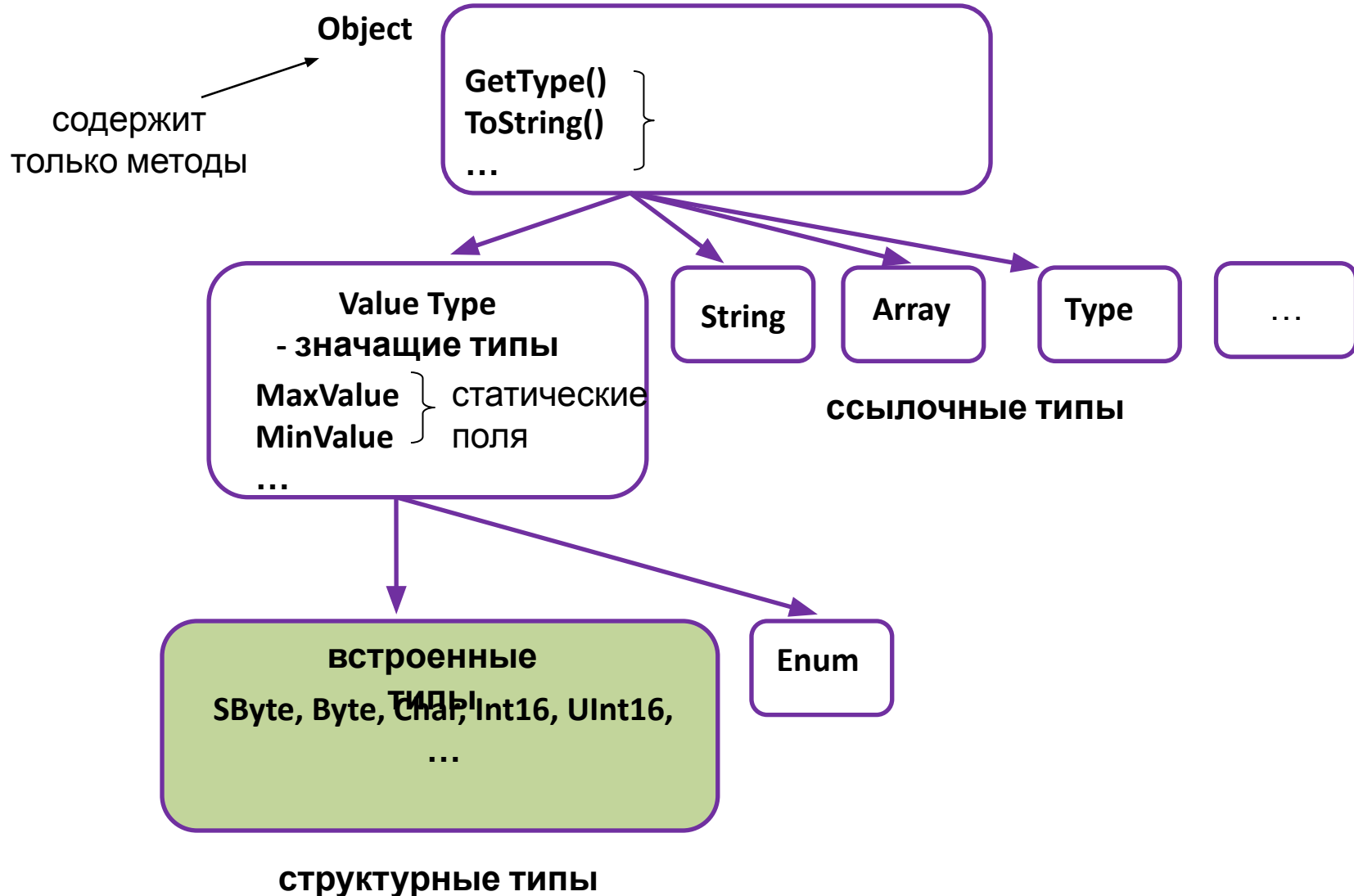
*Хорошую читабельность имени придают знаки нижнего подчёркивания:
my_first_variable
или использование прописных букв MyFirstVariable*

мой_первый_объект

МояПерваяПеременная

Иерархия **встроенных** типов в C#

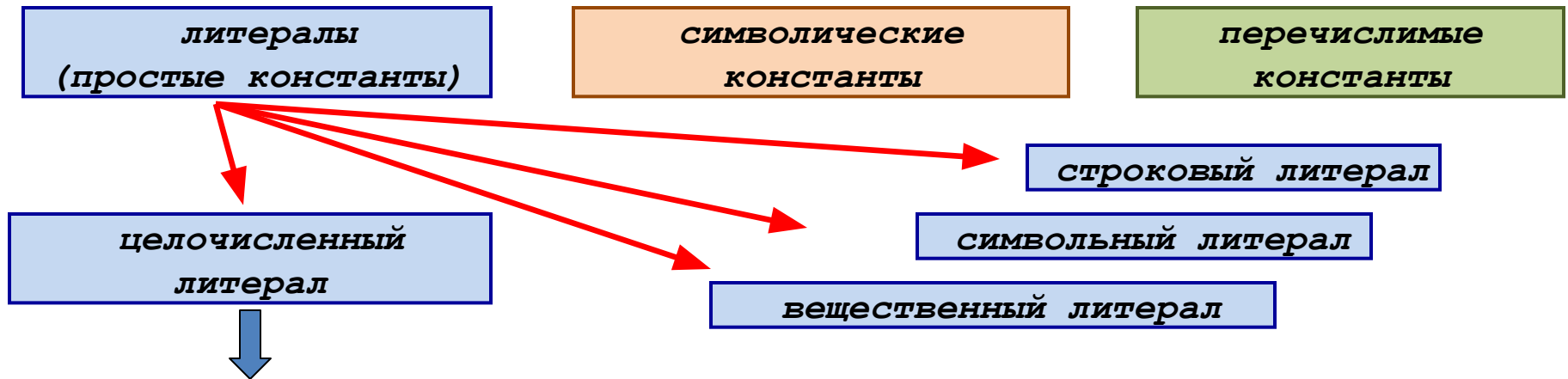
Все типы данных в C# являются производными от системного «супербазового» типа *Object*



Константы

Константами называются *объекты данных*, которые *не изменяют* своего значения на всём времени выполнения программы.

Константы в C# бывают трёх типов :



Целочисленный литерал (или **целочисленная константа**) служит для записи **целых** значений и является последовательностью **цифр**. Эта последовательность может предваряться знаком '-', который в данном случае является **операцией** смены знака.

Целочисленный литерал, начинающийся с **0x** или **0X**, воспринимается транслятором, как шестнадцатеричное целое. В этом случае он может включать ещё и символы от **A** (или **a**), до **F** (или **f**)

Целочисленный литерал

В зависимости от значения целочисленный литерал размещается последовательно в одном из следующих форматов (по мере увеличения значения)

целочисленный знаковый размером 4 байта $\leq 32\,767$

целочисленный беззнаковый размером 4 байта $\leq 65\,535$

целочисленный знаковый размером 8 байт $\leq 2^{63}-1$

целочисленный беззнаковый размером 8 байт $\leq 2^{64}-1$

Непосредственно *за константой* могут располагаться в произвольном сочетании один или два специальных суффикса:

U (или u) и *L (или l)*

использование беззнакового формата

использование формата удвоенного размера – 8 байт

Вещественный литерал

Запись десятичного числа в естественной форме (разделитель целой и дробной части - **точка**)

Запись десятичного числа в экспоненциальной форме (разделитель мантисса/порядок - символ **E (e)**)

Размещается вещественный литерал в **8-байтовом** плавающем формате (соответствует плавающему типу с удвоенной точностью - **double**)

Непосредственно за константой могут располагаться один из двух специальных суффиксов:
F (или f) и **M (или m)**

Использование формата одинарной точности - **float** размером 4 байта

Использование формата повышенной точности - **decimal** размером 16 байт

Символьный литерал



Одна или несколько литер, заключенных в ~~значении~~, является соответствующий код

(Unicode)

Размещается в поле размером **два**

байта



- символ, заключённый в апострофы, например, `'D'`. (печатный символ)

- целочисленная константа после обратного слеша. (для кодов, которые не представлены на клавиатуре, при отсутствии символического представления). Например, константа `'\x44'` содержит код буквы D (шестнадцатеричный)


- заключенная в апострофы литера после символа обратного слеша, т.н. `esc` - последовательности, команды управления некоторыми устройствами:

```
'\u` - задание 4-х значного шестнадцатеричного кода в системе Unicode;  
'\a` - звуковой сигнал (\u0007);  
'\b` - возврат на одну позицию назад (\u0008);  
'\n` - переход на новую строку (\u000A);  
'\r` - возврат каретки ( курсора ) в первую позицию строки (\u000D);  
'\t` - переход к следующей метке горизонтальной табуляции (\u0009);  
'\v` - переход к следующей метке вертикальной табуляции (\u000B);  
'\0` - конец строки, нуль - символ (\u0000), терминатор);  
'\` - апостроф;  
'\"` - кавычки;
```

Пример вывода на консоль целочисленных и символьных констант

```
using System;
class Primer
{ static void Main()
  {
    Console.WriteLine ( 68 );
    Console.WriteLine ( 0x44 );
    Console.WriteLine ('D');
    Console.WriteLine ("\x44");
    Console.WriteLine ( 0x44.ToString() ); // константа – это тоже объект!
  }
}
```

Статические методы класса Console :
WriteLine() – вывод строки на консоль
с переводом курсора на новую строку ;
Write() – то же самое, но без перевода курсора);
ReadLine() – ввод строки с консоли;
Read() – чтение с консоли одного символа;



68
68
D
D
68