Открытые Решения_

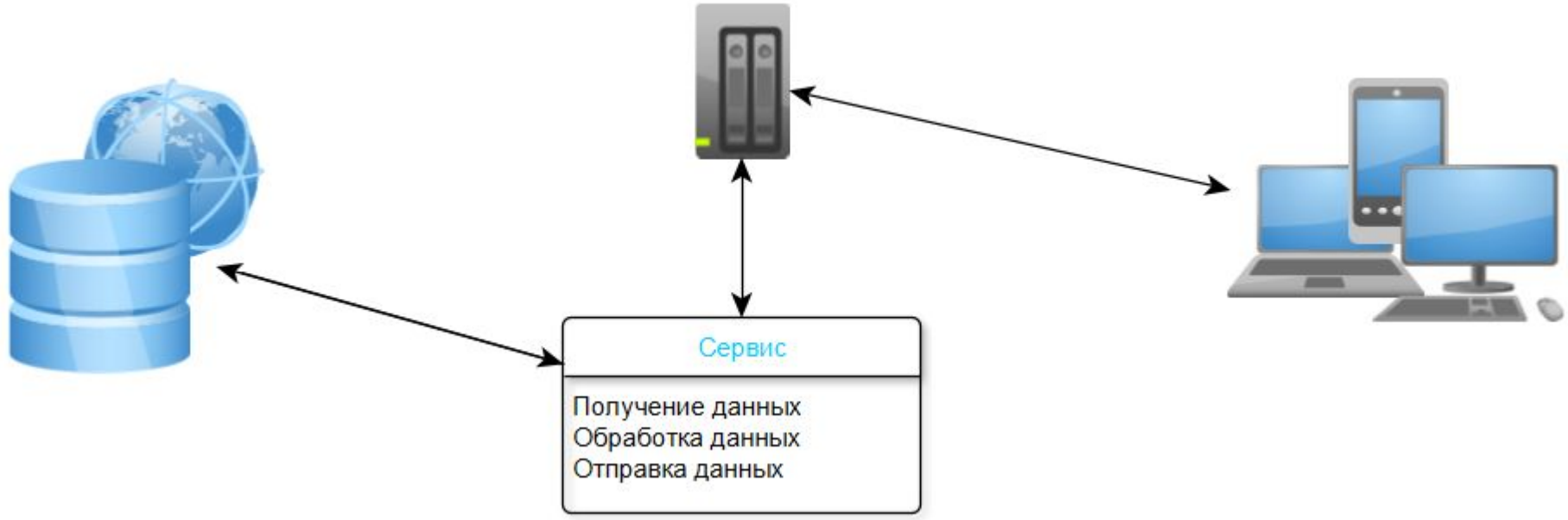
разработка программного обеспечения

Типы и структуры данных в .Net

osinit.com

Открытые Решения_
разработка программного обеспечения

Важная составляющая программирования – обработка данных.



Открытые Решения_
разработка программного обеспечения

В .Net типы данных делятся на две категории:

Значимые

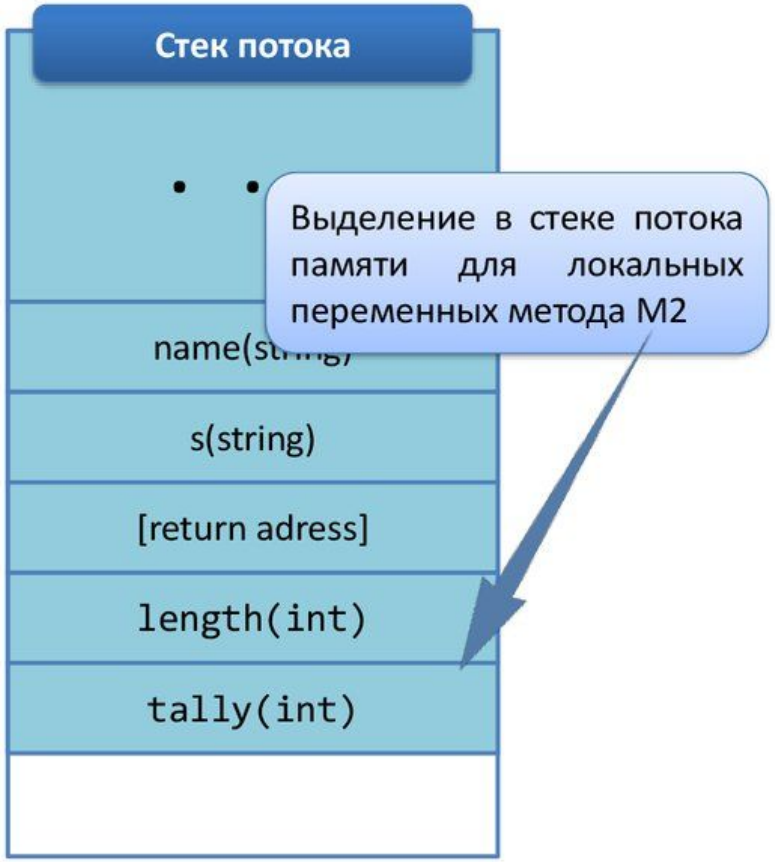
Ссылочные

Открытые Решения_
разработка программного обеспечения

Значимые типы хранятся в стеке потока

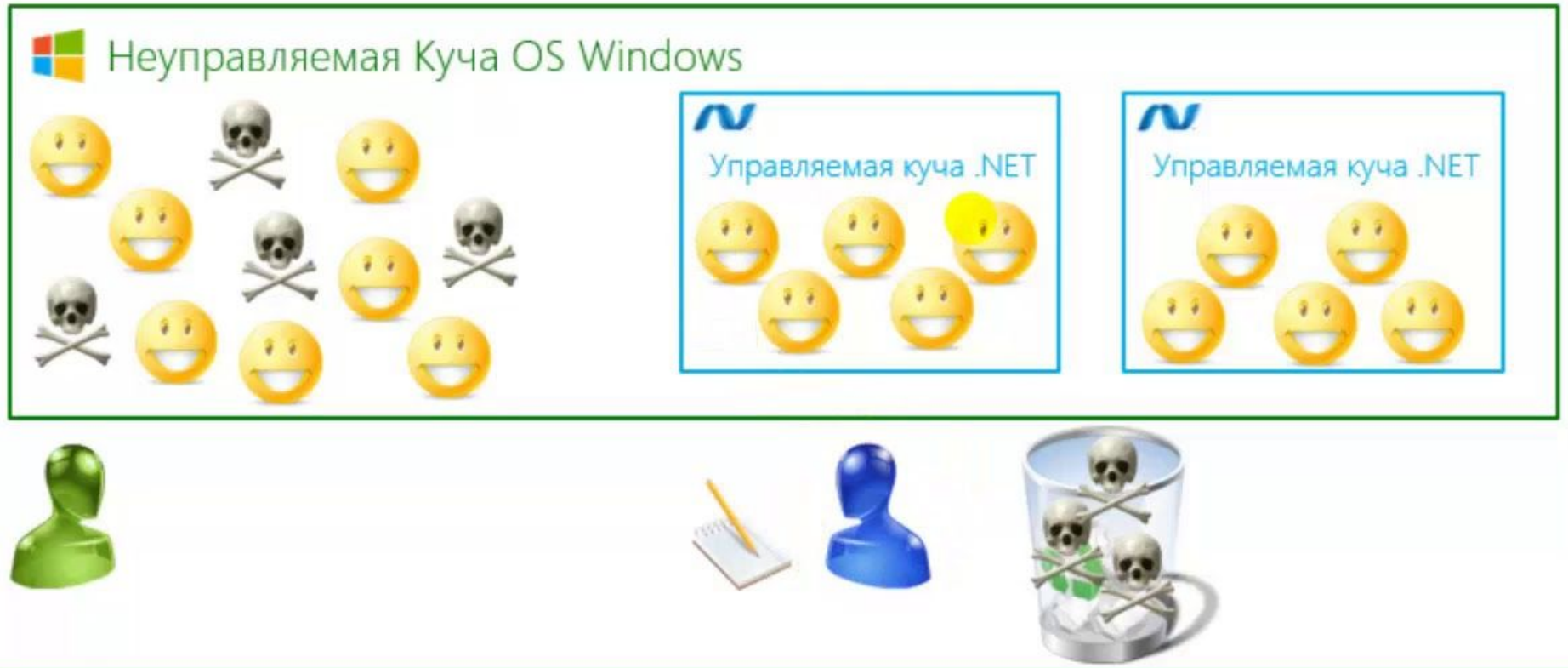
```
void M1()
{
    string name = "Joe";
    M2(name);
    . . .
    return;
}
```

```
void M2(string s)
{
    int length = s.Length;
    int tally;
    . . .
    return;
}
```



Открытые Решения_
разработка программного обеспечения

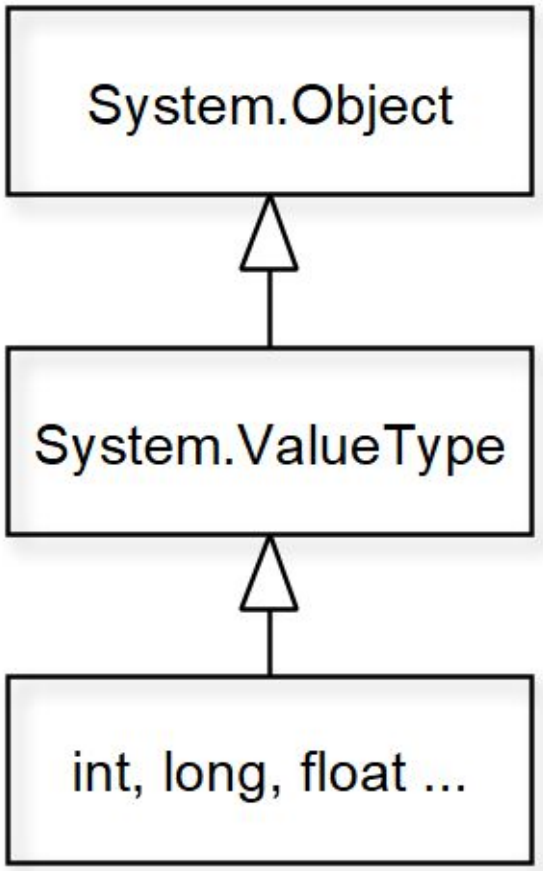
Ссылочные типы хранятся в управляемой куче



Открытые Решения_
разработка программного обеспечения

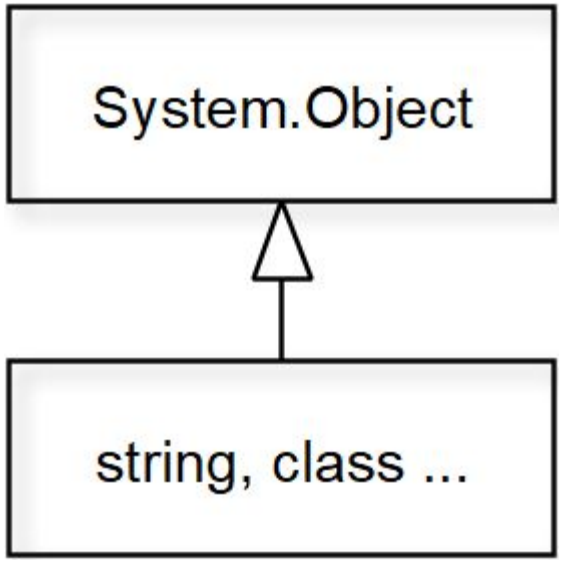
[7]

Значимые



Переопределение Equals

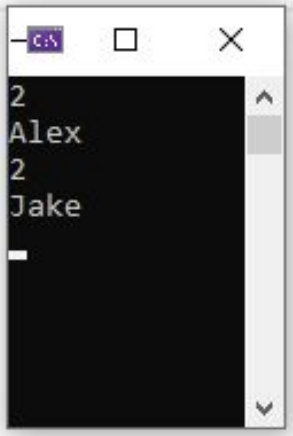
Ссылочные



Открытые Решения_
разработка программного обеспечения

```
namespace OSINITLecture
{
    public class TestClass
    {
        public string Name { get; set; }
    }
}
```

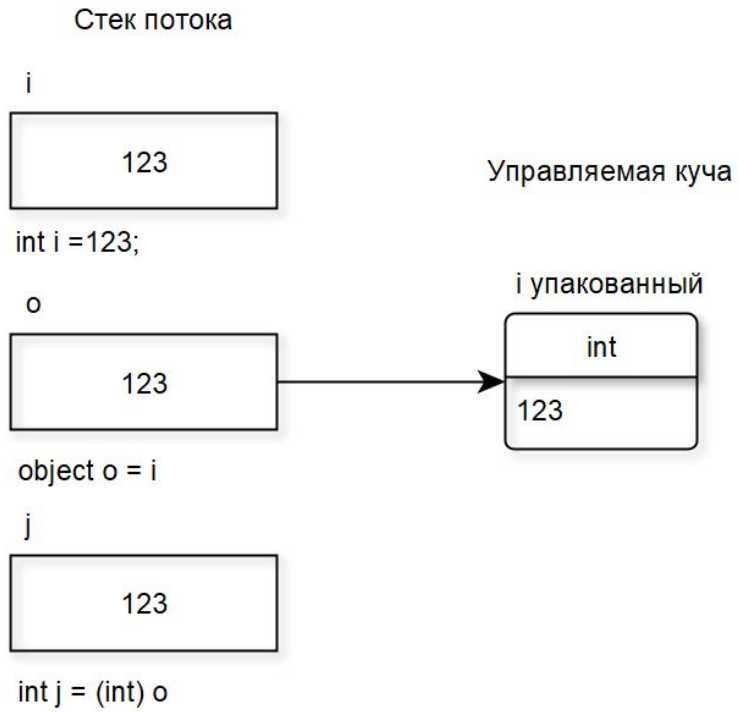
```
using System;

namespace OSINITLecture
{
    class LearnClass
    {
        static void Main(string[] args)
        {
            int number = 2;
            TestClass classObject = new TestClass() { Name="Alex"};

            Console.WriteLine(number);
            Console.WriteLine(classObject.Name);

            Func(number);
            Func(classObject);

            Console.WriteLine(number);
            Console.WriteLine(classObject.Name);

            Console.ReadKey();
        }

        static private void Func(int number)
        {
            number+=10;
        }

        static private void Func(TestClass obj)
        {
            obj.Name = "Jake";
        }
    }
}
```



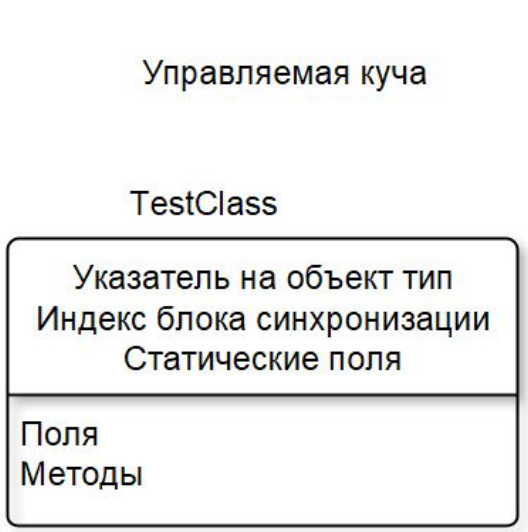
Открытые Решения_
разработка программного обеспечения

[7] Значимые

```
namespace OSINITLecture
{
    class LearnClass
    {
        static void Main(string[] args)
        {
            int i = 123;

            object o = i;

            int j = (int)o;
        }
    }
}
```



Ссылочные



Значимые типы [7]

Целочисленные типы: byte, sbyte, short, ushort, int, uint, long, ulong

```
byte byte1 = 90;
sbyte byte2 = -102;

short s1 = -1102;
ushort s2 = 1102;

int i1 = -1043;
uint i2 = 32000;

long a = -102345520;
ulong c = 23474868;
```

Тип	Разрядность в битах	Диапазон
byte	8	0:255
sbyte	8	-128:127
short	16	-32768 : 32767
ushort	16	0 : 65535
int	32	-2147483648 : 2147483647
uint	32	0 : 4294967295
long	64	-9223372036854775808 : 9223372036854775807
ulong	64	0 : 18446744073709551615

Значимые типы [7]

Типы с плавающей запятой: float, double,decimal

```
float floatNumber = 34656.21123123F;

double doubleNumber = 747126.454624124124456D;

decimal decimalNumber = 1234.37621767367216371267354352525466 2M;
```

Тип	Разрядность в битах	Диапазон
float	32 бита	5E-45 до 3,4E+38
double	64 бита	5E-324 до 1,7E+308
decimal	128 бит	1E-28 до 7,9E+28

Открытые Решения_
разработка программного обеспечения

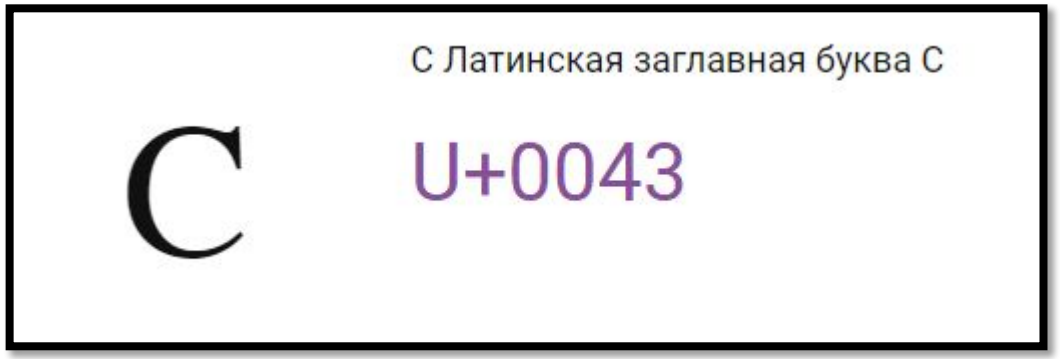
Логический тип: bool

```
bool firstCondition = true;

bool secondCondition = false;
```

Открытые Решения_
разработка программного обеспечения

Символьный тип: char

```
char a = 'A';

char b = '\x62'; //ASCI

char c = '\u0043'; //UNICODE
```



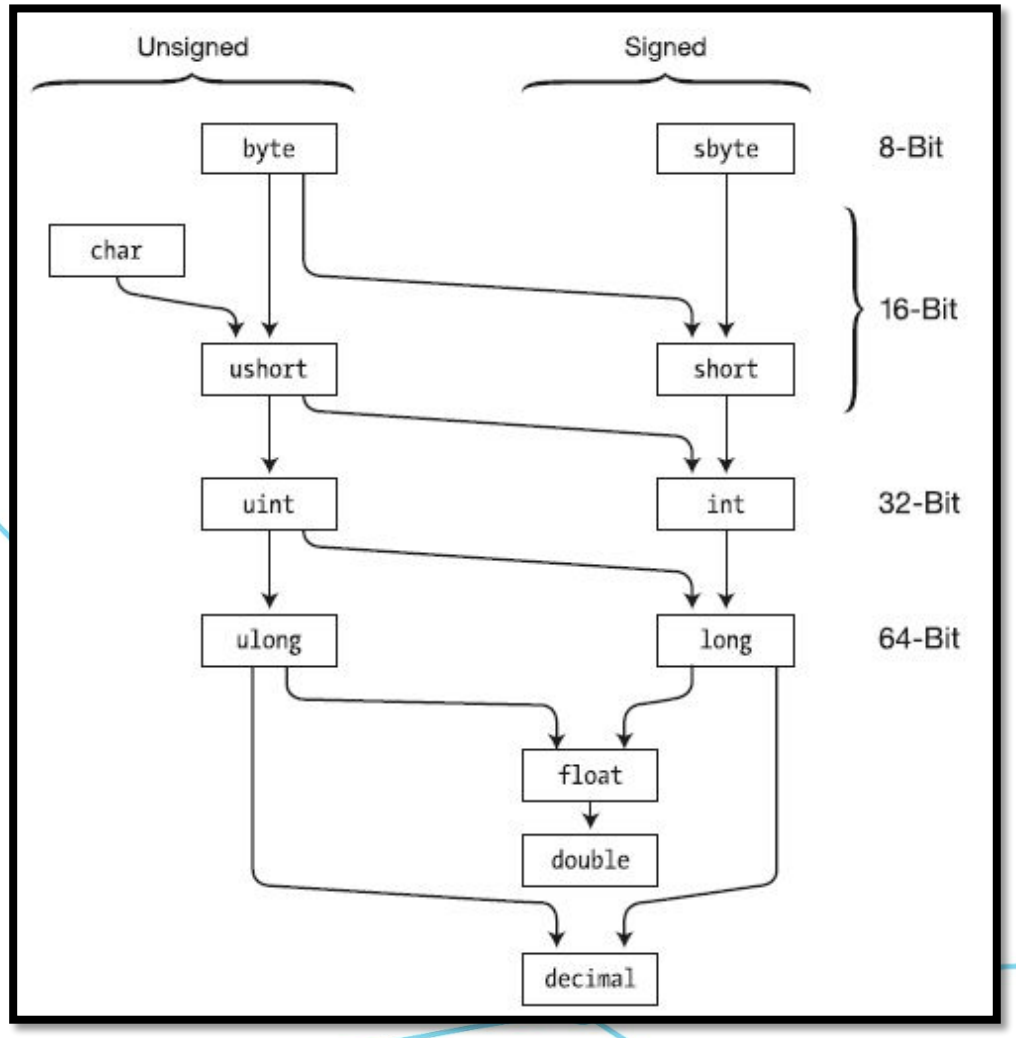
Значимые типы [7]

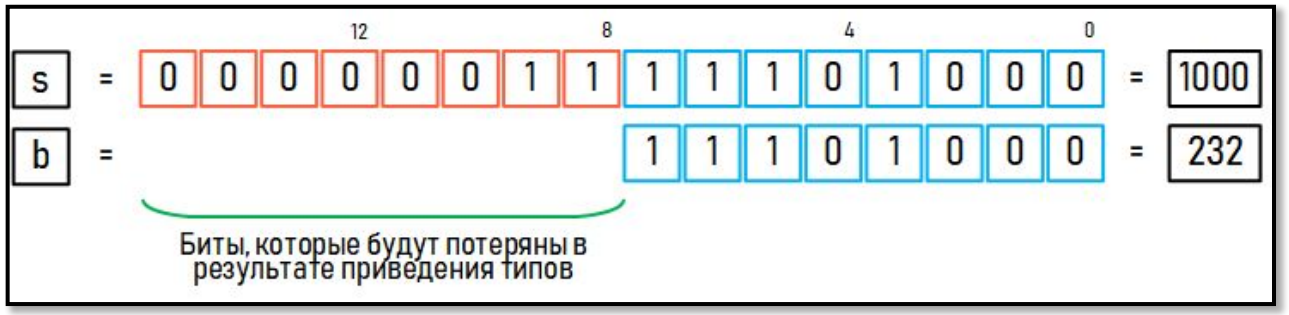
Составной тип: struct

```
struct Car
{
    public string model;
    public int yearOfRelease;

    public void DisplayInfo()
    {
        Console.WriteLine($"Model: {model}  Year Of Release: {yearOfRelease}");
    }
}

public class Trains
{
    public string Name { get; set; }

    public void TestMethod()
    {
        Name = "Test";
    }
}
```

Открытые Решения_
разработка программного обеспечения

Особенности преобразования значимых типов





Значимые типы [7]

Составной тип: enum

```
public enum Materials
{
    Steel = 625123,
    Plastic = 71245,
    Gold = 575987,
    Silver = 72163
}
```

```
public void WriteMaterial(Materials materials)
{
    switch (materials)
    {
        case Materials.Steel:
            Console.WriteLine("Это сталь");
            break;

        case Materials.Plastic:
            Console.WriteLine("Это пластик");
            break;

        case Materials.Gold:
            Console.WriteLine("Это золото");
            break;

        case Materials.Silver:
            Console.WriteLine("Это серебро");
            break;

        default:
            Console.WriteLine("Код метала неизвестен");
            break;
    }
}
```

Ссылочные

Строковая переменная: string

```
string str = "Пример строковой переменной";
```

Compare: сравнивает две строки с учетом текущей локали пользователя
CompareOrdinal: сравнивает две строки без учета локали
Contains: определяет, содержится ли подстрока в строке
Concat: соединяет строки
CopyTo: копирует часть строки, начиная с определенного индекса в массив
EndsWith: определяет, совпадает ли конец строки с подстрокой
Format: форматирует строку
IndexOf: находит индекс первого вхождения символа или подстроки в строке
Insert: вставляет в строку подстроку
Join: соединяет элементы массива строк
LastIndexOf: находит индекс последнего вхождения символа или подстроки в строке
Replace: замещает в строке символ или подстроку другим символом или подстрокой
Split: разделяет одну строку на массив строк
Substring: извлекает из строки подстроку, начиная с указанной позиции
ToLower: переводит все символы строки в нижний регистр
ToUpper: переводит все символы строки в верхний регистр
Trim: удаляет начальные и конечные пробелы из строки

Ссылочные

Класс

Класс – структура данных, описывающая объект предметной области.

Как правило, класс состоит из свойств (полей) содержащих данные и методов – функций меняющих эти данные.

```
Cars MyCar = new Cars("Cheveolet Lanos", 170);
```

```
public class Cars
{
    public string Model { get; set; }
    public double MaxSpeed { get; set; }
    public double CurentSpeed { get; set; }

    public Cars(string model, double maxSpeed)
    {
        Model = model;
        MaxSpeed = maxSpeed;
        CurentSpeed = 0;
    }

    public void Ride()
    {
        CurentSpeed = 60;
    }

    public void Stop()
    {
        CurentSpeed = 0;
    }
}
```

Ссылочные

Интерфейс

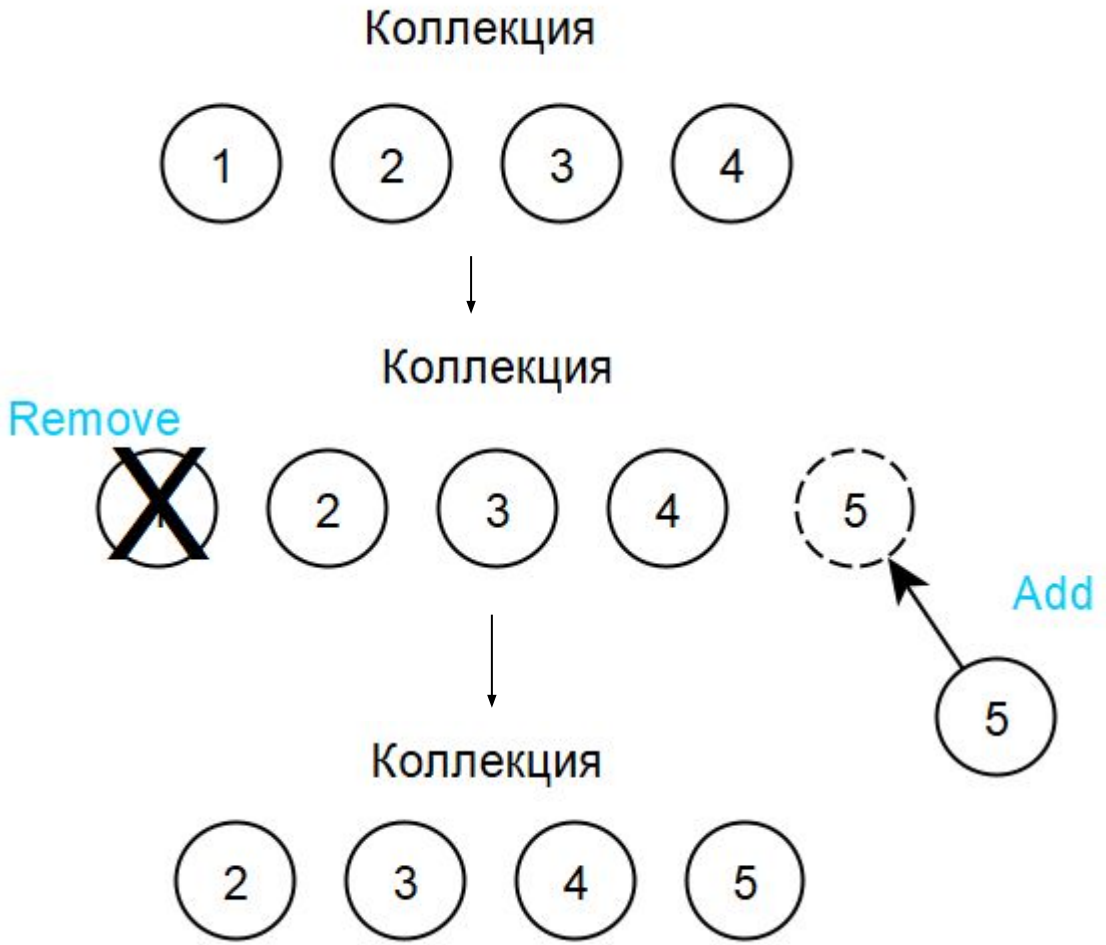
Интерфейс содержит перечень методов, которые должны быть реализованы в классе.

```
public interface IPassengerCar
{
    public void GetPassenger();
    public void DropPassenger();
}
```

```
public class Cars : IPassengerCar
{
    public string Model { get; set; }
    public double MaxSpeed { get; set; }
    public double CurentSpeed { get; set; }

    public Cars(string model, double maxSpeed)
    {
        Model = model;
        MaxSpeed = maxSpeed;
        CurentSpeed = 0;
    }

    public void Ride()
    {
        CurentSpeed = 60;
    }

    public void Stop()
    {
        CurentSpeed = 0;
    }

    public void GetPassenger()
    {
        Console.WriteLine("Берем пассажира");
    }

    public void DropPassenger()
    {
        Console.WriteLine("Высаживаем пассажира");
    }
}
```

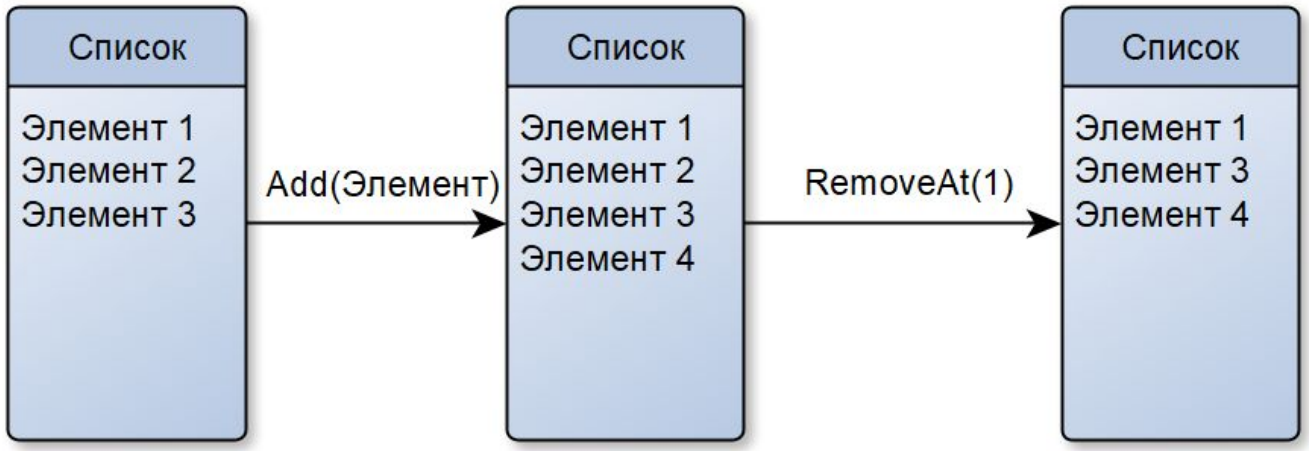
Открытые Решения_
разработка программного обеспечения

Абстрактные типы данных



Открытые Решения_
разработка программного обеспечения

Список - List

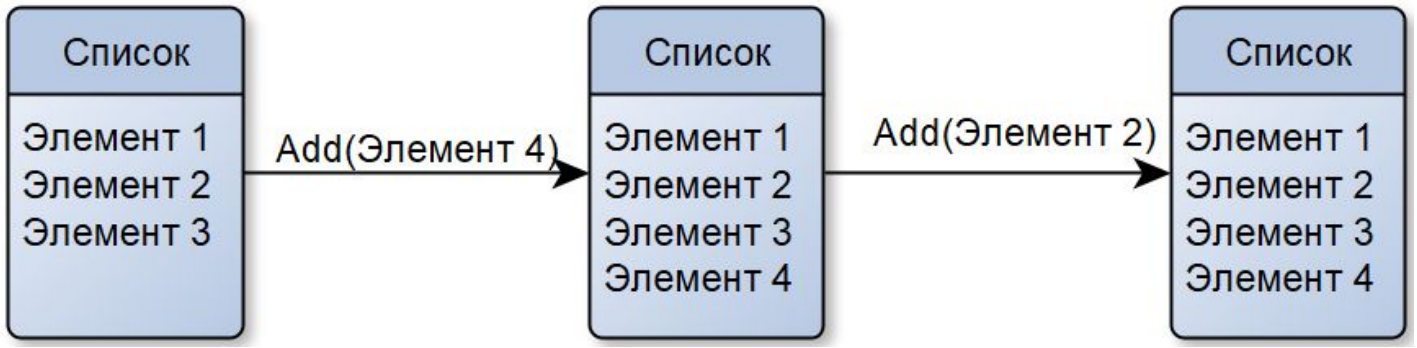



```
List<Cars> Autos = new List<Cars>();
```

```
Cars MyCar = new Cars("Chevrolet Lanos", 170);
Autos.Add(MyCar);
```

Открытые Решения_
разработка программного обеспечения

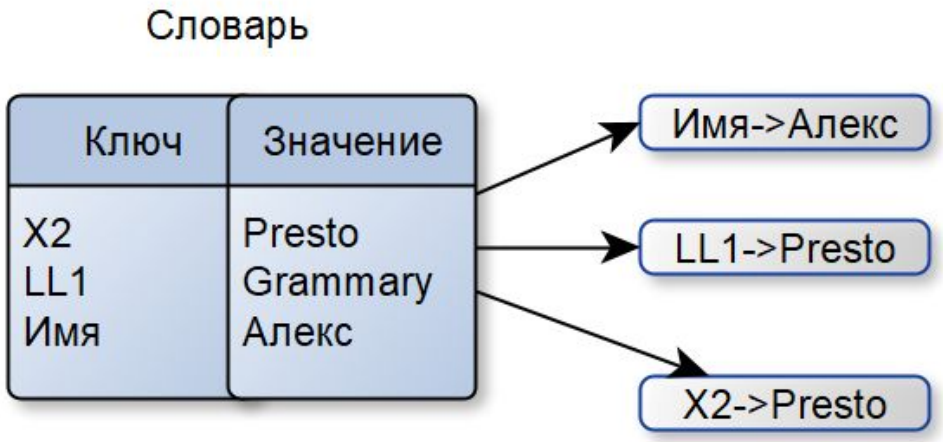
Уникальный список - ISet




```
ISet<int> set = new HashSet<int> { 1, 2, 3, 4, 5 };
set.Add(0); //{0, 1, 2, 3, 4, 5 }
set.Add(1); //{0, 1, 2, 3, 4, 5 }
```

Открытые Решения_
разработка программного обеспечения

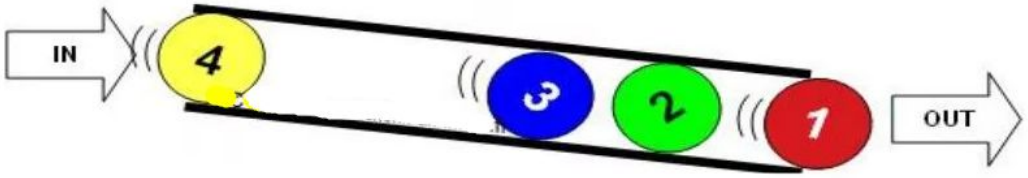
Словарь - Dictionary



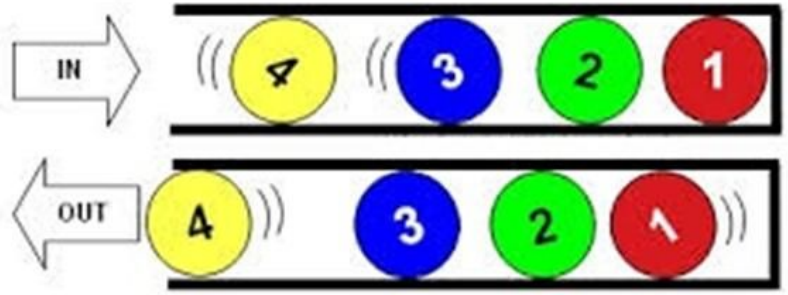

```
Dictionary<string, int> Marks = new Dictionary<string, int>();

Marks.Add("отлично", 5);
Marks.Add("хорошо", 4);
Marks.Add("удовлетворительно", 3);

Console.WriteLine(Marks["отлично"]); // 4
```

Открытые Решения_
разработка программного обеспечения

Очередь - Queue и Стек - Stack



Queue



Stack

```
var Queue = new Queue<int>(new[] { 1, 2, 3, 4, 5 });

Queue.Enqueue(6); //{ 1, 2, 3, 4, 5, 6 }
var dequeuedItem = Queue.Dequeue();// 1
```

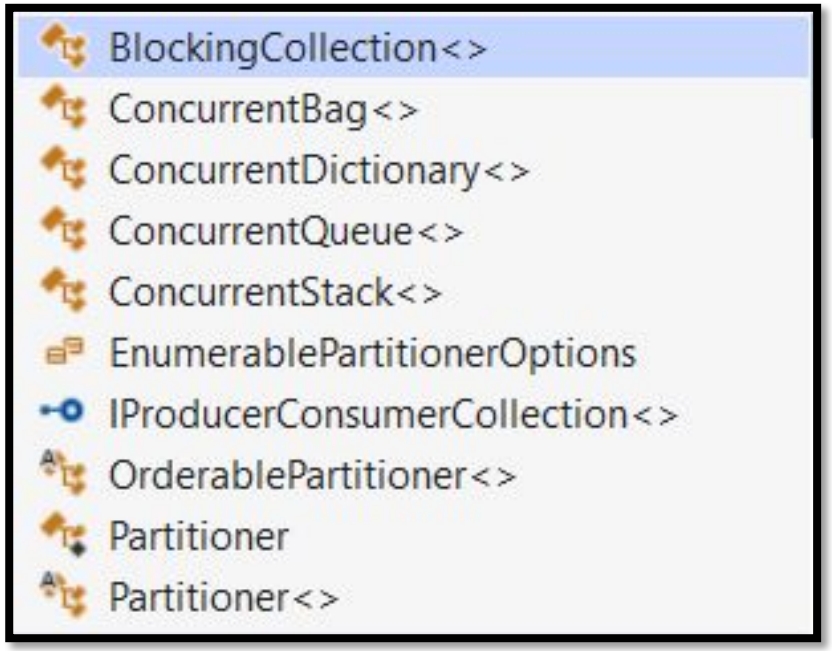
```
var Stack = new Stack<int>(new[] { 1, 2, 3, 4, 5 });

Stack.Push(6);//{ 1, 2, 3, 4, 5 , 6 }
var poppedItem = Stack.Pop(); //{ 1, 2, 3, 4, 5}
var peekedItem = Stack.Peek(); //5
```

Открытые Решения_
разработка программного обеспечения

Concurrent collections

Пространство имен **System.Collections.Concurrent** содержит коллекции, которые являются потокобезопасными и специально предназначены для параллельного программирования.

Это означает, что они могут безопасно использоваться в многопоточной программе, где возможен одновременный доступ к коллекции со стороны двух или больше параллельно исполняемых потоков и минимизируют возможные ситуации некорректно записи данных в коллекции.

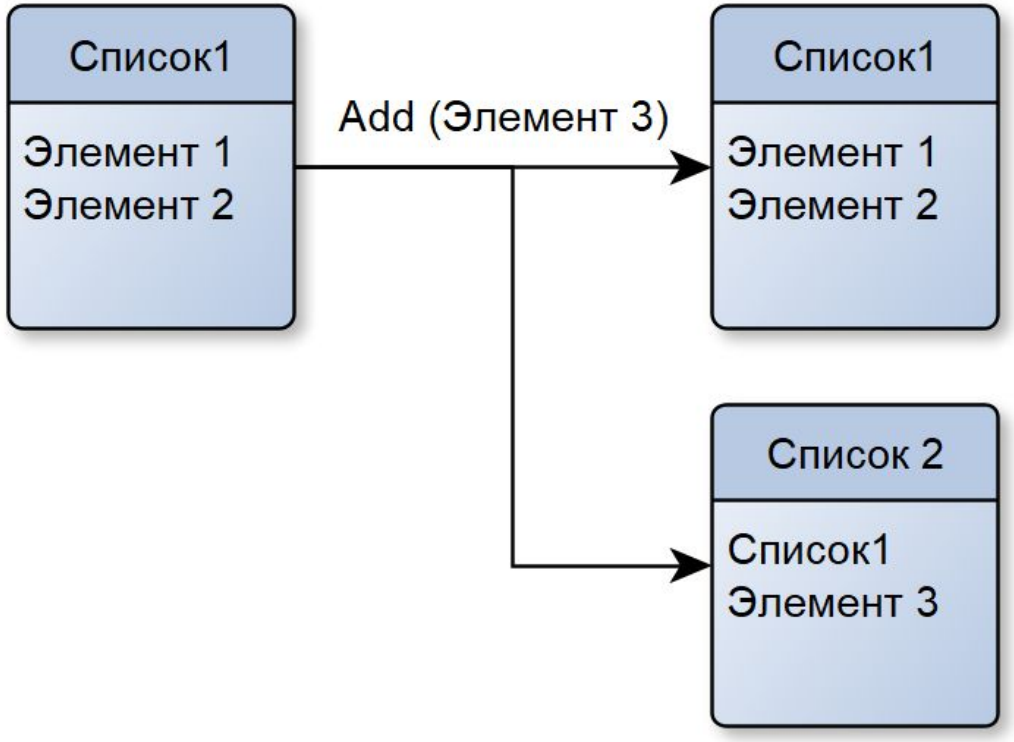


Открытые Решения_
разработка программного обеспечения

Immutable collections – неизменяемые коллекции

Неизменяемые коллекции не могут быть изменены после их создания.

Все операции, которые обычно изменяют коллекцию, возвращают новый экземпляр.



Открытые Решения_
разработка программного обеспечения

Как правильно выбрать тип данных?

Исходя из контекста решаемой задачи:

- Если предполагается работа только с целыми числами то можно выбрать тип int
- Если числа не могут быть отрицательными то используем беззнаковые типы, например uint
- Важен порядок обработки – используем коллекцию Queue
- Число объектов заранее неизвестно – используем List<>

И т.д.

- Однопоточные сценарии: классические коллекции
- Запись из множества потоков: concurrent коллекции, защищающие внутреннее состояние и имеющие подходящее для конкурентной записи API

Открытые Решения_

разработка программного обеспечения

Спасибо за внимание

440000, Россия, г. Пенза, ул. Московская, д. 29
+7(8412) 200-604
info@osinit.com

osinit.com