Web-разработка на C# и платформе Microsoft .NET

Основные конструкции

План занятия

- Типы данных
- Концепции ввода/вывода
- Ключевые операции

Виды типов объектов

VALUE TYPE

- Находятся в стеке
- Передаются по значению
- Будут существовать в области видимости локальной переменной
- Содержат сами данные

REFERENCE TYPE

- Находятся в куче
- Передаются по ссылке
- Уничтожаются сборщиком мусора
- Содержат в себе ссылку на данные

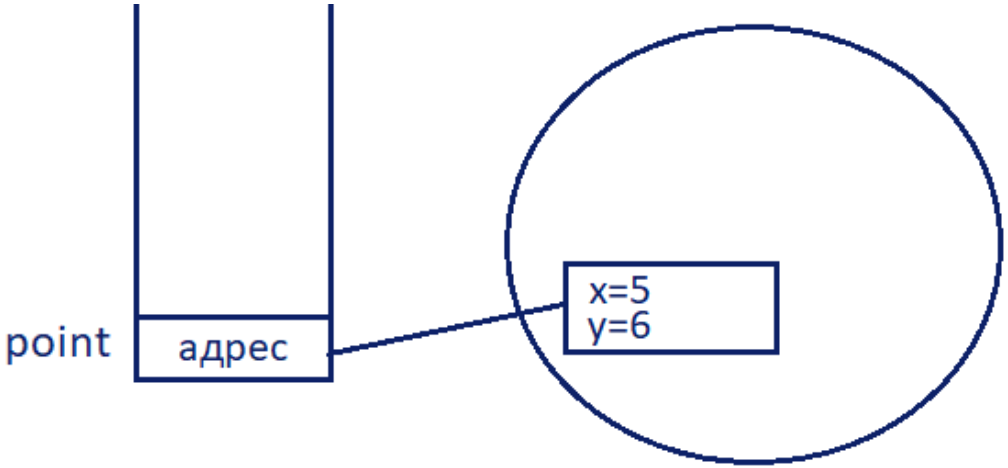
Виды памяти

СТЕК (СТАТИЧЕСКАЯ ПАМЯТЬ)

```
int x = 5;
```

КУЧА (ДИНАМИЧЕСКАЯ ПАМЯТЬ)

```
MyPoint point = new MyPoint();
point.X = 5;
point.Y = 6;
```



Простые value типы

- Целые: **int**, byte, short, long
- Вещественные: float, **double**
- Повышенной точности: decimal
- Логический: bool
- Символьный: char

Value типы, которые может создать пользователь

СТРУКТУРЫ

```
struct MyPoint
{
    public int x { get; set; }
    public int y { get; set; }
}

MyPoint p = new MyPoint();
p.x = 5;
p.y = 6;
```

ПЕРЕЧИСЛЕНИЯ

```
enum DaysOfWeek
{
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
}

DaysOfWeek day =
    DaysOfWeek.Monday;
```

Приведение значимых типов

- Неявное приведение (автоматически) – от «меньшего» к «большему»

  `char -> int -> long -> float -> double`

- Явное приведение (вручную) – от «большего» к «меньшему»

  `double -> float -> long -> int -> char`

- Оператор «is» – используется для проверки типов перед их приведением, помогает избегать ошибок, связанных с преобразованием типов

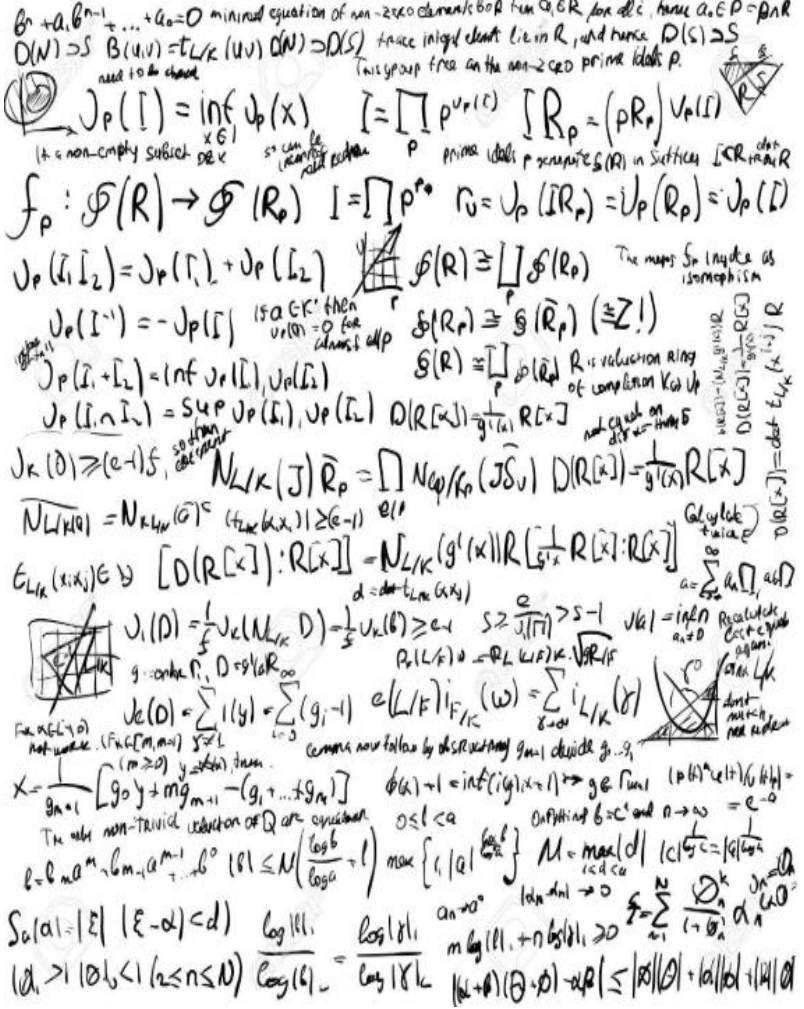
```
if (student is User)
{
    var stud = (User)student;
    Console.WriteLine(stud.Name);
}
```

- Оператор «as» - практически идентичен «is», но выполняет еще и приведение, а не только проверку

```
var stud = new Student() as User;
```

Операции

- Арифметические: + - / * %
- Сравнения: > < >= <= == !=
- Логические: && || ! & |
- Битовые: & | ~ ^
- Присваивания: = += -= /= *= %= ++ -- &= |=



Операторы языка C#

- **Условные**: `if, switch, ?:, ??`
- **Циклы**: `while, do/while, for, foreach`
- **Перехода**: `break, continue, goto, return`

```
var st:User = student is User
    ? student.Name != reserved
        ? student.Id != 0
            ? student.status == "preselected"
                ? student
                : new Student(student)
            : new Student(student)
        : new Student(student)
    : new User(student);
```

Оператор if

```
if(условие)
{
  действия, если условие
  истинно
}
else
{
  действия, если условие
  ложно
}
```

```
if (a > b)
{
    max = a;
}
else
{
    max = b;
}
```

Оператор switch

```
switch ( <выражение> )
{
  case <константное_выражение_1>:
     [<оператор 1>;]  <оператор перехода>;
  case <константное_выражение_2>:
     [<оператор 2>;] <оператор перехода>;
   …
  case <константное_выражение_n>:
     [<оператор n>;] <оператор перехода>;
  [default:
     [<оператор>;] <оператор перехода>;]
}
```

Оператор switch

```
string str = Console.ReadLine();
int n;
switch (str)
{
    case "Понедельник": n = 5;  break;
    case "Вторник":     n = 4;  break;
    case "Среда":       n = 3;  break;
    case "Четверг":     n = 2;  break;
    case "Пятница":     n = 1;  break;
    default:            n = 0;  break;
}
Console.WriteLine("Осталось работать {0} дней", n);
```

Оператор for

for([<инициализация>]; [<выражение>]; [<модификация>])
{
** тело цикла**
}

```
Console.Write("N =");
string str = Console.ReadLine();
int n = int.Parse(str);
for (int i = 1; i <= n; i++)
{
    Console.Write(" {0}", i);
}
Console.ReadLine();
```

Оператор while

while (условие)
{
** тело цикла**
}

```
Console.Write("N =");
string str = Console.ReadLine();
int n = int.Parse(str);
int i = 1;
while (i <= n)
{
    Console.Write(" {0}", i++);
}
Console.ReadLine();
```

Оператор do while

do
{
тело цикла
}while(условие)

```
Console.Write("N =");
string str = Console.ReadLine();
int n = int.Parse(str);
int i = 1;
do
{
    Console.Write(" {0}", i++);
} while (i <= n);
Console.ReadLine();
```

Выбор типа цикла

- Вывести на экран числа от 1 до 100.
- Применить действие ко всем элементам массива.
- Вывести на экран первые 100 строк файла.
- Подсчитать сумму бесконечного ряда с заданной точностью.

Вывод данных

Отдельные значения

```
Console.WriteLine(x);
```

Ручное объединение строк

```
Console.WriteLine("x = " + x + ", y = " + y);
```

Форматирование

```
Console.WriteLine("x = {0}, y = {1}", x, y);
```

Ввод данных

- Ввод данных с клавиатуры возможен только в строковом формате;

- Для приведения строки к конкретному типу, как правило, используются методы Parse и TryParse;

- При ошибке преобразования типа может произойти исключительная ситуация, однако предусматривать её не обязательно.

```
Console.Write("N = ");

string str = Console.ReadLine();
int n = int.Parse(str);

Console.WriteLine("Your input: {0}", n);
Console.ReadLine();
```

Блок try – catch – finally

Синтаксис:

```
try
{
        действия_способные_привести
        к_исключительной_ситуации
}
catch [(тип_исключения [переменная_исключения])]
{
        обработка_исключения
}
[finally
{
        действия_выполняемые_вне_зависимости
        от_результата_блока_try
}]
```

Блок try – catch – finally

Синтаксис:

```
try
{
        действия_способные_привести
        к_исключительной_ситуации
}
catch [(тип_исключения [переменная_исключения])]
{
        обработка_исключения
}
[finally
{
        действия_выполняемые_вне_зависимости
        от_результата_блока_try
}]
```

Обработка неверного ввода перехватом исключения

```
string str = Console.ReadLine();

try
{
    int n = int.Parse(str);
}
catch (FormatException)
{
    Console.WriteLine("Wrong input!");
    return;
}
```

СПАСИБО ЗА ВНИМАНИЕ