ТИПЫ ДАННЫХ

C# - ЛЕКЦИЯ 3

НА ПРОШЛОМ ЗАНЯТИИ

- Структура программы
- Абстракции языка: по данным и по действиям
- Конструкции языка: инструкции и выражения, ветвления и цилы

БУДЕМ РАССМАТРИВАТЬ ПОЗЖЕ

- Контроль переполнения: checked/uncheked
- Обработка ошибок: try-catch
- Работа с ресурсами: using
- Организация критических секций: lock
- Итераторы: yield
- Pattern matching: switch
- Анонимные функции

ТИПЫ ДАННЫХ

ОБЩЕПРИНЯТОЕ ОПРЕДЕЛЕНИЕ ТИПА ДАННЫХ

1. Объем занимаемой памяти
2. Способ представления в памяти
3. Диапазон значений
4. Набор операций

ТИПЫ ДАННЫХ В C#

1. Классы
2. Структуры
3. Перечисления
4. Интерфейсы
5. Делегаты

КЛАССЫ

Это ссылочные типы, описывающие объекты

На один экземпляр класса может указывать множество ссылок

Ссылка может иметь неопределённое значение - `null`

СТРУКТУРЫ

ПЕРЕЧИСЛЕНИЯ

ИНТЕРФЕЙСЫ

ДЕЛЕГАТЫ

ЧЛЕНЫ ТИПА

- Конструкторы
- Деструктор
- Поля
- Свойства
- Методы
- Константы
- Индексаторы
- События
- Операторы

КОНСТРУКТОРЫ

- Описывает инициализацию экземпляра типа после его создания
- Конструктор гарантированно отрабатывает при создании каждого экземпляра типа
- Конструктор отрабатывает не более одного раза
- Перегрузка конструкторов - объявление более одного конструктора для одного типа. Позволяет описать различные сценарии инициализации.

КОНСТРУКТОРЫ

```
//  конструктор без параметров структуры System.DateTime
DateTime defaultDate = new DateTime();

//  конструктор с параметрами: год, месяц, число
DateTime birthDate = new DateTime(1989, 5, 9);
```

ДЕСТРУКТОР

- Описывает деинициализацию экземпляра перед его уничтожением
- Вызывается автоматически сборщиком мусора
- Недоступен для использования

Поля

- Реализуют абстракцию по данным
- Обычно недоступны сторонним типам
- Доступ к полям обычно организуется посредством свойств

СВОЙСТВА

- Подобны полям для стороннего наблюдателя
- Разделяют сценарии чтения и записи
- Позволяют выполнить дополнительные действия при чтении/записи неявно для выполняющего чтение/запись
- Позволяет запретить чтение (read-only) или запись (write-only)
- Позволяет реализовать "вычисляемые значения": fullname = firstname + middlename + lastname

СВОЙСТВА

```
//  свойство Year структуры System.DateTime
DateTime birthDate = new DateTime(1989, 5, 9);
int year = birthDate.Year;

//  свойства X и Y структуры System.Drawing.Point
Point p = new Point();
p.X = 7;
p.Y = 3;
```

МЕТОДЫ

- Реализуют абстракцию по действиям
- Могут принимать аргументы каких-либо типов
- Могут возвращать результат какого-либо типа

МЕТОДЫ

```
TimeSpan time1 = new TimeSpan(3, 5, 16);  //  3 ч  5 мин 16 с
TimeSpan time2 = new TimeSpan(1, 17, 29); //  1 ч 17 мин 29 с

//  метод Add структуры System.TimeSpan
TimeSpan sum = time1.Add(time2);          //  4 ч 22 мин 45 с
```

КОНСТАНТЫ

- Неизменяемые значения
- Значения констант известны на этапе компиляции программы на C#

КОНСТАНТЫ

```
//  константа PI класса System.Math
double pi = Math.PI;
```

ИНДЕКСАТОРЫ

- Позволяют использовать с экземпляром типа квадратные скобки
- В квадратные скобки передаются аргументы
- В результате - значение какого-либо типа

ИНДЕКСАТОРЫ

```
string str = "Hello World!";

//  Индексатор класса System.String
char ch = str[6];
```

СОБЫТИЯ

- Подписка на событие - передача экземпляру с событием ссылки на метод
- Этот метод будет вызван когда возникнет некоторое событие

СОБЫТИЯ

```
//  подписка на событие Elapsed класса System.Timers.Timer
Timer timer = new Timer(1000);
timer.Elapsed += HandleTimeElapsed;
timer.Start();
```

ОПЕРАТОРЫ

- Унарные
- Бинарные
- Приведения типа

УНАРНЫЕ ОПЕРАТОРЫ

```
bool a = true;
bool b = !a;    //  унарный оператор "!"

int i = 5;
int j = 7;
int k = i + j;  //  бинарный оператор "+"
```

ОПЕРАТОРЫ ПРИВЕДЕНИЯ ТИПА

```
float f = 124.146f;
int i = (int)f;   //  явное приведение типа float к типу int

SomeType t = new SomeType();
int a = t;        //  неявное приведение некоторого типа к типу int
```

ЧЛЕНЫ ТИПОВ

- Члены экземпляров - индивидуальны для каждого экземпляра
- Члены типов (статические) - не относятся к экземплярам

СТАТИЧЕСКИЕ ЧЛЕНЫ ТИПОВ

```
class Program
{
    static void Main()  //  статический метод
    {
        DateTime now = DateTime.Now; //  статическое свойство
    }
}
```

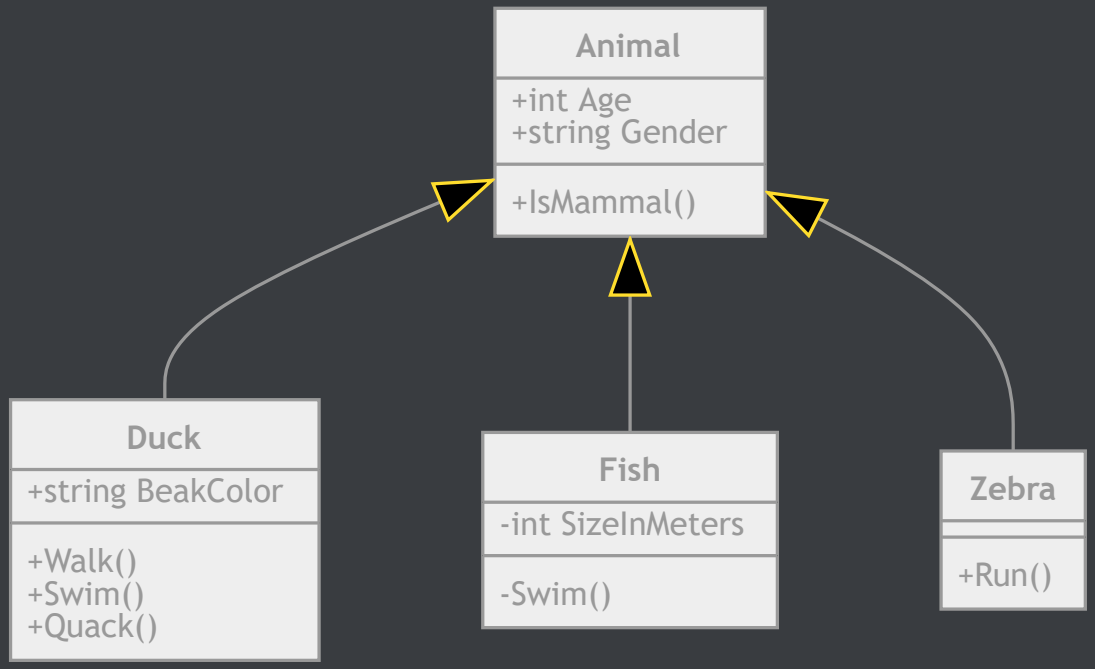
ЧЛЕНЫ ЭКЗЕМПЛЯРОВ ТИПОВ

```
class Program
{
    static void Main()  //  статический метод
    {
        DateTime now = new DateTime(1989, 5, 9);     //  конструктор экземп
        DateTime after20years = now.AddYears(20);    //  метод экземпляра
    }
}
```

НАСЛЕДОВАНИЕ

- Пусть тип A наследует от типа B
- Тогда экземпляры A обладают тем же набором членов, что и экземпляры B, а также членами, которые объявлены в классе A
- A - наследник B, а B - базовый класс для A

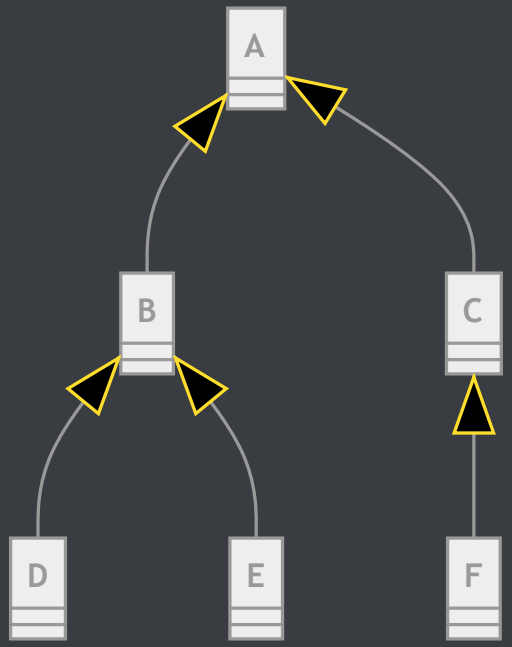
НАСЛЕДОВАНИЕ



ПРИВОДИМОСТЬ ТИПОВ

- Если A приводим к типу B, то значение типа A можно присвоить переменной или передать в виде аргумента типа B
- Типы приводимы по цепочке наследования в сторону базового типа

ПРИВОДИМОСТЬ ТИПОВ



ПРИВОДИМОСТЬ ТИПОВ

```
A a1 = new A();
A a2 = new B();
A a3 = new C();
B b1 = new E();
A a4 = b1;
```

ВСТРОЕННЫЕ В CLI ТИПЫ

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Byte	byte	1 Б	0...255	
System.SByte	sbyte	1 Б	—128...127	

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Int16	short	2 Б	$-2^{15}...2^{15}-1$	
System.UInt16	ushort	2 Б	$0...2^{16}-1$	

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Int32	int	4 Б	$-2^{31}...2^{31}-1$	
System.UInt32	uint	4 Б	$0...2^{32}-1$	

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Int64	long	8 Б	$-2^{63}...2^{63}-1$	L
System.UInt64	ulong	8 Б	$0...2^{64}-1$	UL

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Single	float	4 Б	$\pm 1{,}5 \times 10^{-45} \ldots \pm 3{,}4 \times 10^{38}$	f, F
System.Double	double	8 Б	$\pm 5 \times 10^{-324} \ldots \pm 1{,}7 \times 10^{308}$	d, D
System.Decimal	decimal	16 Б	$\pm 1 \times 10^{-28} \ldots \pm 7{,}9228 \times 10^{28}$	m, M

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Boolean	bool	1 Б	{true, false}	

Полное имя типа	Псевдоним	Объём памяти	Диапазон значений	Константный литерал
System.Char	char	2 Б	\U+0000...\U+FFFF	'a'

СТЕК И КУЧА

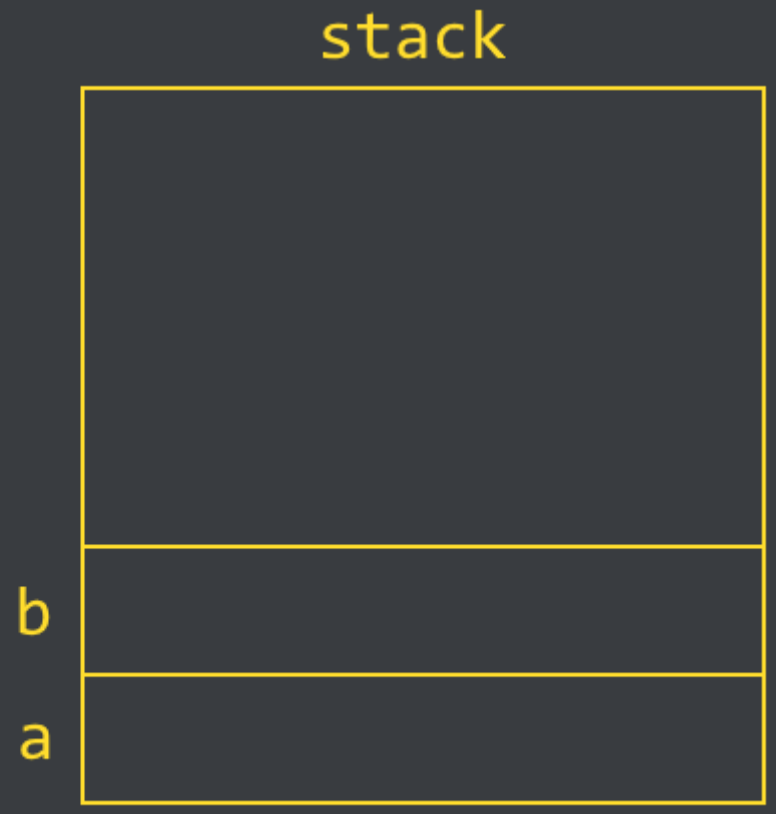
СТЕК

- Хранит локальные переменные
- Хранит значения аргументов вызова текущего метода
- Хранит адрес места вызова текущего метода
- При выходе из области видимости со стека снимаются все её переменные

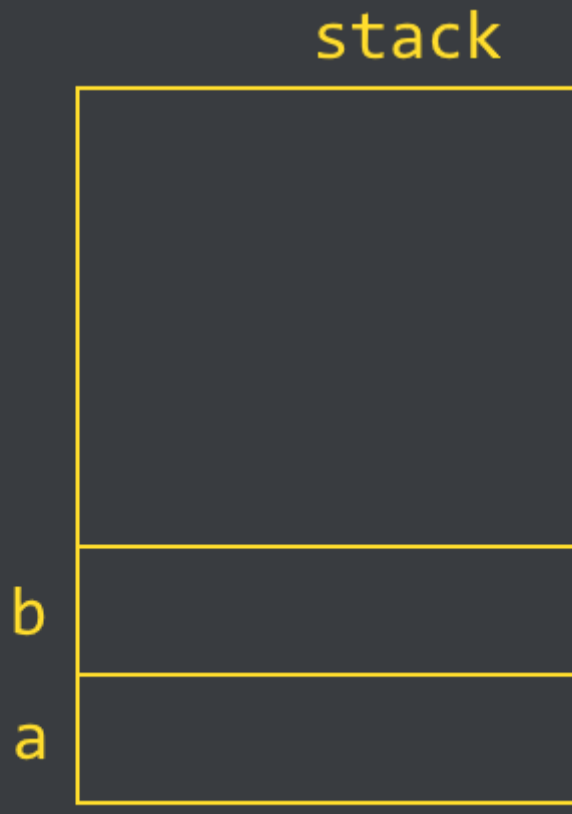
СТЕК

- Имеет малый размер
- Быстрый доступ
- Автоматически очищается
- Не может быть фрагментирован
- Хранит только локальные переменные

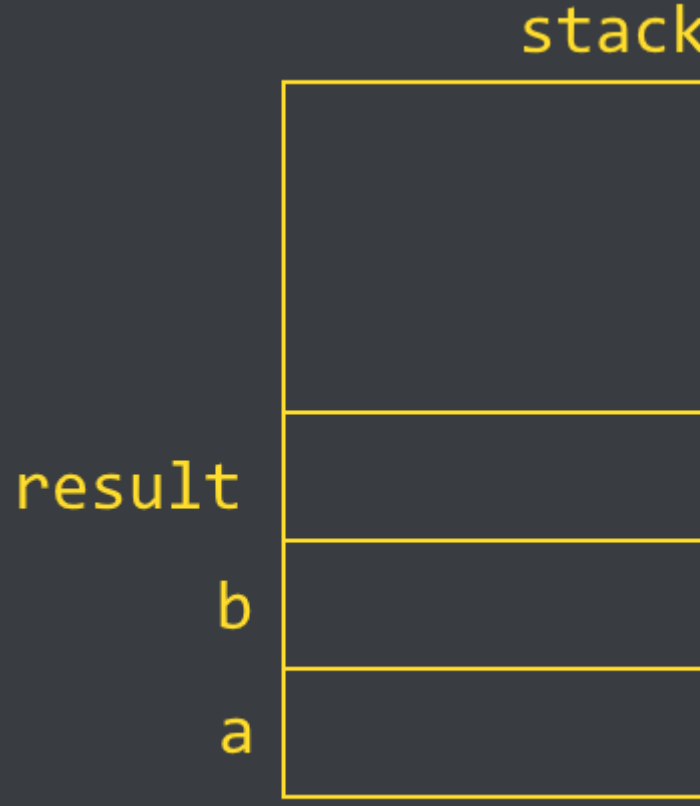
```
ass Program

  static void Main()
  {
      double a, b;
  }
```



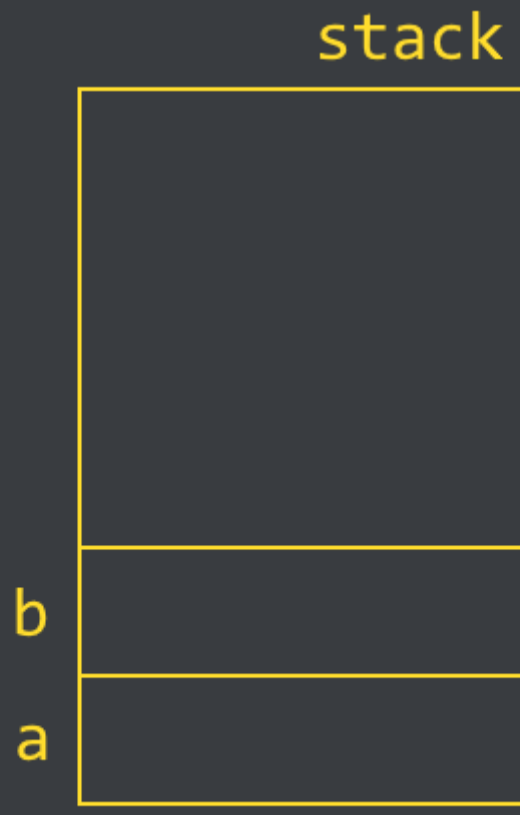
```
ass Program

  static void Main()
  {
      double a, b;

      a = double.Parse(Console.ReadLine());
      b = double.Parse(Console.ReadLine());
  }
```



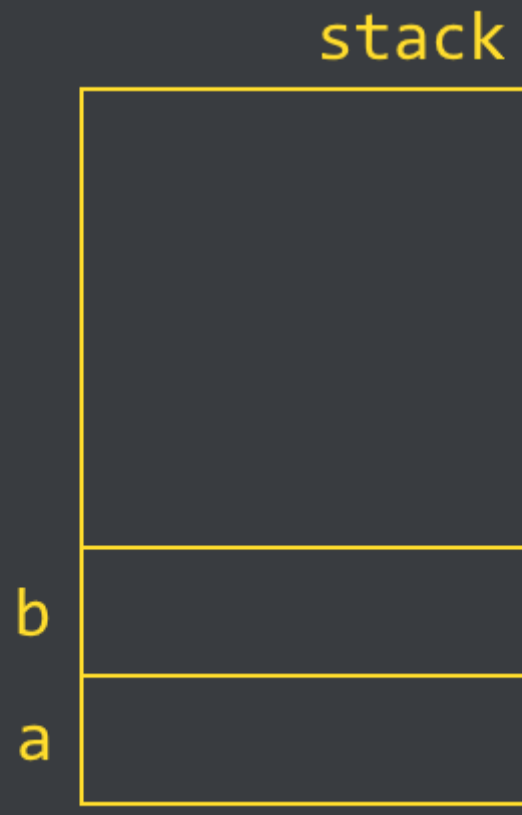
```
 Program

tatic void Main()

   double a, b;

   a = double.Parse(Console.ReadLine());
   b = double.Parse(Console.ReadLine());

   double result = a + b;

   Console.WriteLine($"{a} + {b} = {result}");
```



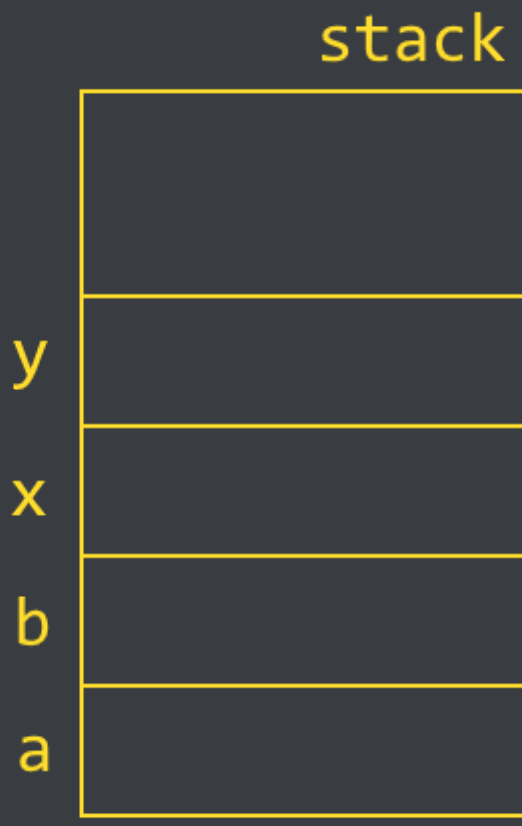
```
atic void Main()

  double a, b;
  --> execution here
  a = double.Parse(Console.ReadLine());
  b = double.Parse(Console.ReadLine());

  double result = Sum(a, b);

  Console.WriteLine($"{a} + {b} = {result}");


atic double Sum(double x, double y)

  return x + y;
```



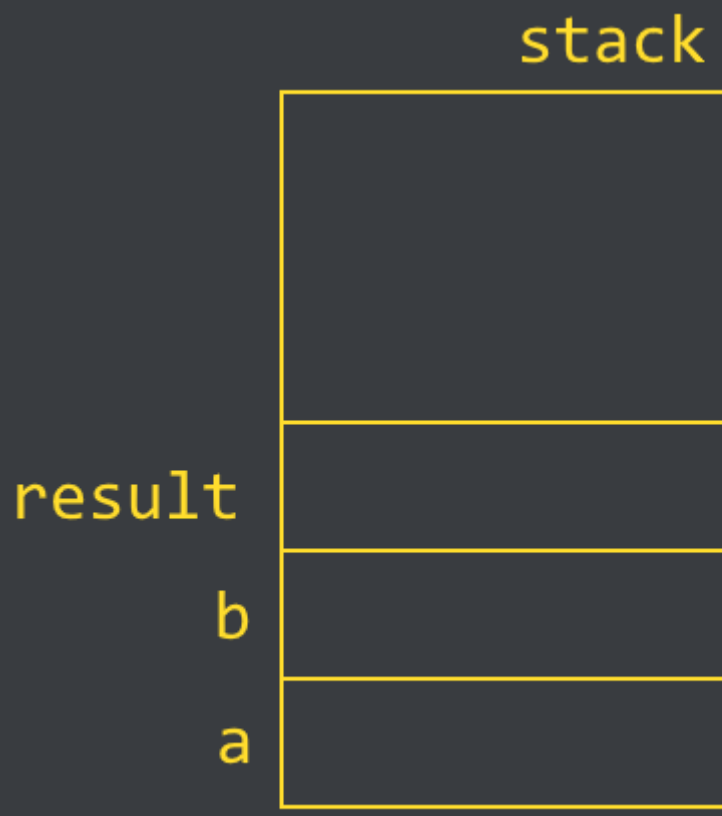
```
atic void Main()

  double a, b;

  a = double.Parse(Console.ReadLine());
  b = double.Parse(Console.ReadLine());
  //  --> execution here
  double result = Sum(a, b);

  Console.WriteLine($"{a} + {b} = {result}");


atic double Sum(double x, double y)

  return x + y;
```



```
atic void Main()

  double a, b;

  a = double.Parse(Console.ReadLine());
  b = double.Parse(Console.ReadLine());

  double result = Sum(a, b);

  Console.WriteLine($"{a} + {b} = {result}");


atic double Sum(double x, double y)

  --> execution here
  return x + y;
```



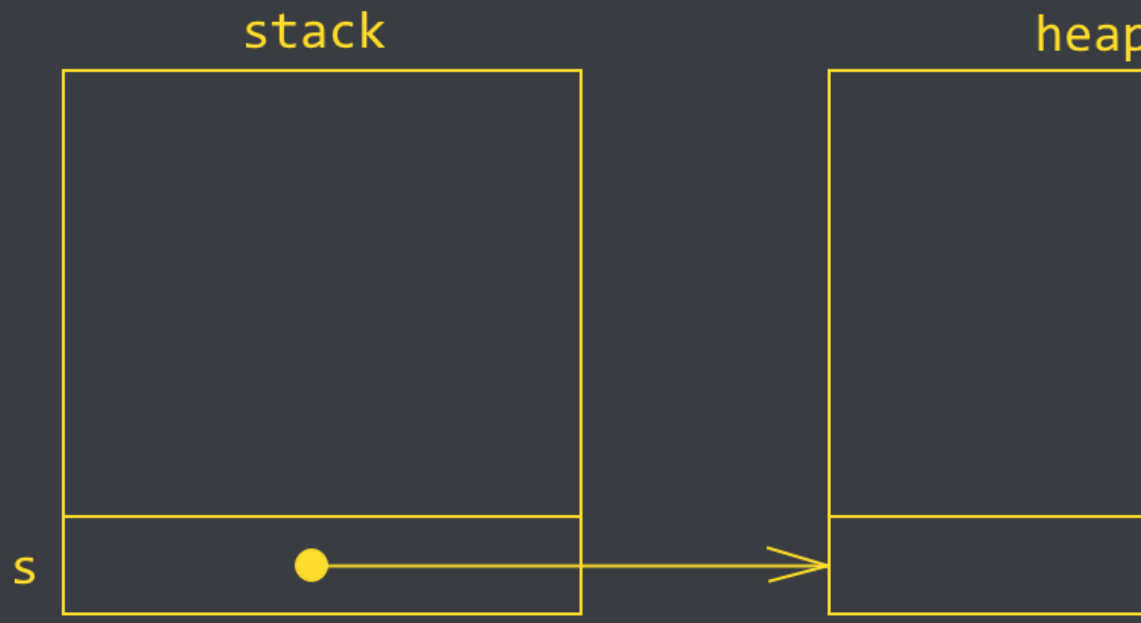
```
atic void Main()

  double a, b;

  a = double.Parse(Console.ReadLine());
  b = double.Parse(Console.ReadLine());

  double result = Sum(a, b);
  //  --> execution here
  Console.WriteLine($"{a} + {b} = {result}");


atic double Sum(double x, double y)

  return x + y;
```



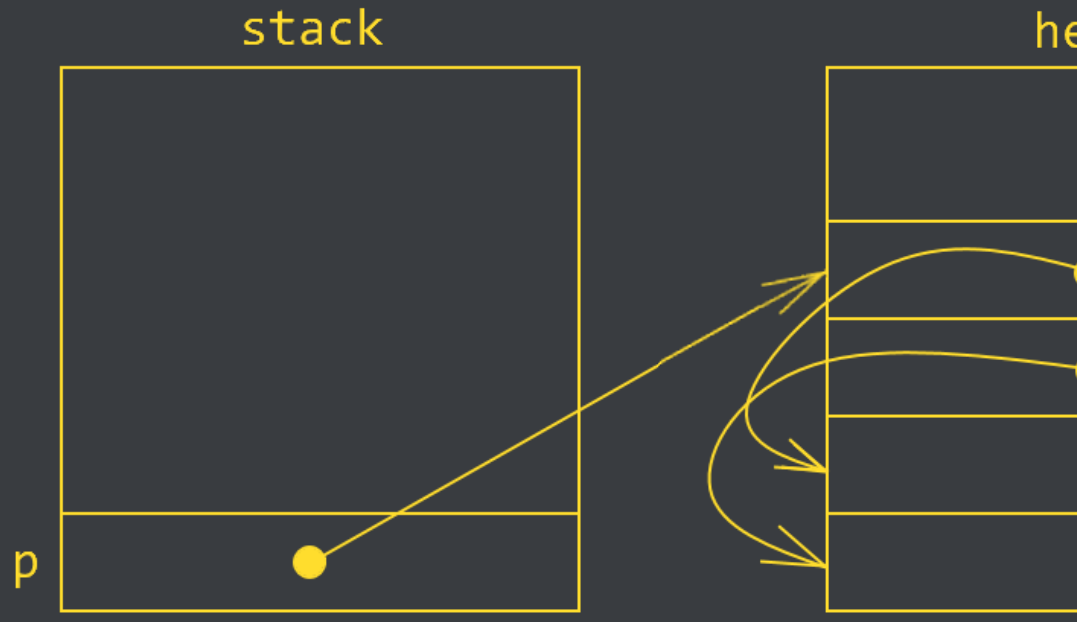
КУЧА

- Не ограничена в размере
- Более медленный доступ
- Очищается сборщиком мусора
- Дефрагментируется сборщиком мусора
- Время жизни данных не зависит от области видимости

```
atic void Main()

  string s = "hello";
```



```
atic void Main()

  Person p = new Person();
  p.Surname = "Jackson";
  p.Name = "Michael";
```



ССЫЛОЧНЫЕ ТИПЫ И ТИПЫ ЗНАЧЕНИЙ

ТИПЫ ЗНАЧЕНИЙ

- Хранятся на стеке
- Копируются значения при присваивании и передаче в качестве аргумента
- Значение по умолчанию неотличимо от остальных: `default(int) == 0`

ССЫЛОЧНЫЕ ТИПЫ

- Хранятся в куче
- Копируются ссылки при присваивании и передаче в качестве аргумента
- Значение по умолчанию - особое: `default(string) == null`

ТИПЫ ЗНАЧЕНИЙ

- Структуры
- Перечисления

ССЫЛОЧНЫЕ ТИПЫ

- Классы
- Интерфейсы
- Делегаты

ВОПРОСЫ