Uvod u C#

Katedra za informatiku
Fakultet tehničkih nauka
Univerzitet u Novom Sadu

Istorija

- Razvijen u Microsoft-u 2002. godine
- Vođa projekta Anders Hejlsberg
- Razvoj jezika započet 1999. (radni naziv “Cool”)
- Kreiran na tragu C++ i Java jezika
- Jezik publikovan 2002.
- Standardizovan od strane ECMA i ISO/IEC
- Trenutna verzija 9.0 (iz novembra 2020)

Generalne osobine jezika

- Jezik opšte namene
- Objedinjuje više programskih paradigmi
  - Stroga tipiziranost
  - Strukturiranost
  - Objektna orijentisanost
  - Imperativnost
  - Deklarativnost
  - Generičnost
- Slobodan format kucanja programskog koda

Identifikatori

- Dozvoljeno je koristiti
  - Velika i mala slova
  - Cifre (identifikator ne sme počinjati cifrom)
  - Karakter _
- Kao identifikator se ne mogu koristiti rezervisane reči:

abstract	do	in	protected	true
as	double	int	public	try
base	else	interface	readonly	typeof
bool	enum	internal	ref	uint
break	event	is	return	ulong
byte	explicit	lock	sbyte	unchecked
case	extern	long	sealed	unsafe
catch	false	namespace	short	ushort
char	finally	new	sizeof	using
checked	fixed	null	stackalloc	virtual
class	float	object	static	void
const	for	operator	string	volatile
continue	foreach	out	struct	while
decimal	goto	override	switch	
default	if	params	this	
delegate	implicit	private	throw	

Promenljive

- Lokacija za skladištenje određene vrednosti
- Konvencija imenovanja
  - camelCase notacija
    - Malo početno slovo, svaka nova reč počinje velikim slovom
- Mora biti promenljiva deklarisana pre korišćenja
  - Navode se tip i naziv promenljive
  - int a;
- Nije moguće koristiti promenljivu pre nego što joj se dodeli i vrednost (*definite assignment rule*)
  - Kompajler će prijaviti grešku

Dodela vrednosti

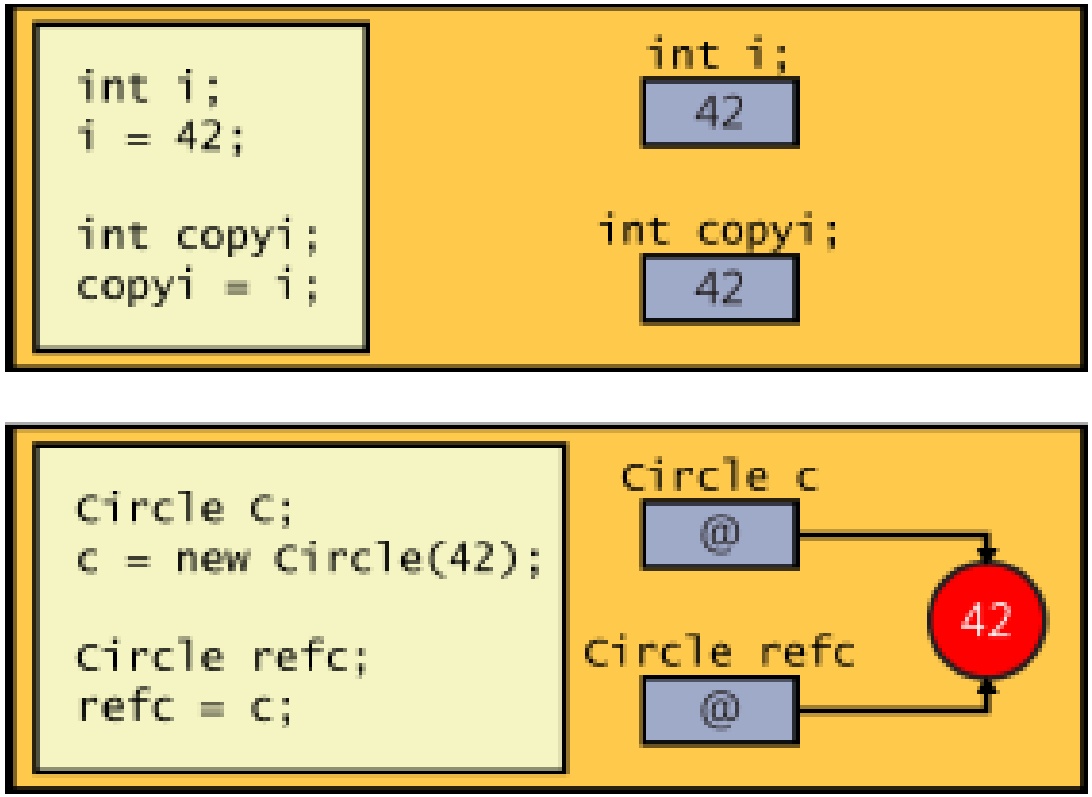
- Znak =

```
cena = 78.99;
```

Tipovi podataka

- Vrednosni tipovi
  - instance se skladište u stek memoriji
- Adresni tipovi
  - adrese instanci se skladište u stek memoriji
  - instance se skladište u heap memoriji
- Garbage Collection
  - oslobađanje memorije koju zauzimaju nekorišćeni objekti
  - mehanizam je pod kontrolom CLR i automatski se aktivira

Reprezentacija u memoriji



Vrednosni tipovi podataka

- Primitivni tipovi

Data type	Description	Size (bits)	Range	Sample usage
int	Whole numbers (integers)	32	-2^{31} through $2^{31} - 1$	int count; count = 42;
long	Whole numbers (bigger range)	64	-2^{63} through $2^{63} - 1$	long wait; wait = 42L;
float	Floating-point numbers	32	$\pm 1.5 \times 10^{-45}$ through $\pm 3.4 \times 10^{38}$	float away; away = 0.42F;
double	Double-precision (more accurate) floating-point numbers	64	$\pm 5.0 \times 10^{-324}$ through $\pm 1.7 \times 10^{308}$	double trouble; trouble = 0.42;
decimal	Monetary values	128	28 significant figures	decimal coin; coin = 0.42M;
char	Single character	16	0 through $2^{16} - 1$	char grill; grill = 'x';
bool	Boolean	8	True or false	bool teeth; teeth = false;

Vrednosni tipovi podataka

- Enumeracija
  - Promenljive ovog tipa mogu dobiti samo vrednost iz skupa diskretnih vrednosti koje enumeracija predviđa
  - Npr. promenljiva koja predstavlja dan u nedelji

```
enum Dan {Ponedeljak, Utorak, Sreda, Cetvrtak, Petak};

Dan d = Dan.Petak;
```

  - Moglo bi se rešiti i korišćenjem int promenljivih, ali bi
    - kod bio manje čitak
    - promenljiva bi mogla da dobije vrednost bilo kojeg celog broja (ne bi bila ograničena na samo nekoliko mogućih vrednosti)

Tipovi podataka

- Enumeracija
  - Vrednosti iz enumeracije se mapiraju na brojeve (počevši od nula)

```
Dan d = Dan.Petak;
int br = (int) Dan.Petak; //br je 4
```

  - Može i eksplicitno da se odredi broj na koji se vrednost mapira

```
enum Dan {Ponedeljak = 3, Utorak = 4,
Sreda = 5, Cetvrtak = 6, Petak = 7};
```

  - Ako se ne odredi broj, uzima se prethodni uvećan za 1

```
enum Dan {Ponedeljak = 3, Utorak, Sreda,
Cetvrtak, Petak};
```

Vrednosni tipovi podataka

- Struktura
  - Sadrži grupisane podatke i operacije (metode) zajedničke za jedan entitet
  - Kao klasa, samo što je vrednosni tip koji se skladišti na steku

```
public struct Test
{
  public int a, b;
}
Test t;
t.a = 10;
t.b = 5;
```

Tipovi podataka

- Adresni tipovi
  - klase, interfejsi, nizovi, delegati
- Vrednosti ovog tipa se skladište u heap memoriji
  - U stek memoriji se skladišti adresa iz heap memorije na kojoj je promenljiva uskladištena
- Svi složeni tipovi izvedeni od tipa `System.Object` sa metodama
  - `ToString`
  - `Equals`
  - `GetHashCode`
  - `Finalize`

Automatsko određivanje tipa

- Kompajler može da automatski odredi tip promenljive na osnovu vrednosti koja se dodeljuje

```
var x = 5;
```

- x će biti tipa int nakon gornjeg izraza
- Ako se koristi **`var`**, mora se odmah izvršiti dodela vrednosti

Konverzije tipova

- Implicitna konverzija u situacijama kada nema gubljenja podataka

```
int a = 10;
double b = a;
```

- Eksplicitna konverzija ako može doći do gubitka podataka

```
double b = 10;
int a = (int) b;
```

String [1]

- Niz karaktera je reprezentovan klasom `System.String`
- Može se koristiti i alias `string`
- Adresni tip
- Operatori `==` i `!=` kreirani tako da porede **vrednosti**, a ne adrese
- Neizmenjiv (pri svakoj dodeli vrednosti zauzima se nova memorija)
- Operator `[]` za preuzimanje karaktera na određenoj poziciji

String [2]

- Postavljanje sadržaja stringa:
  - `string s = ”uvod”;`
  - `string s = “uvod\tstringovi”;`
  - `string s = “c:\\temp\\test”`
  - `string s = @”c:\temp\test”;`
- Dužina stringa
  - `s.Length`
- Spajanje stringova

```
string s1 = “uvod”;
string s2 = “stringovi”;
string s = s1 + “ “ + s2;
```

- Metode
  - `ToLower, ToUpper`
  - `Replace, Insert, Remove`
  - `Substring`
  - `Split`
  - `Trim, TrimLeft, TrimRight`

Aritmetički operatori

- Aritmetički operatori
  - Standardni aritmetički operatori `+`, `-`, `/`, `*`
  - Operator `%` za ostatak pri deljenju
  - `++` i `--` prefiksni i postfiksni operatori za inkrementiranje i dekrementiranje
- Aritmetički sa dodelom vrednosti
  - `+=, -=, *=, /=`

- Postoji posebna vrednost Infinity za decimalni broj koji predstavlja beskonačno
- Postoji posebna vrednost NaN za decimalni broj koji predstavlja nevalidnu vrednost

Relacioni operatori

- Porede dve vrednosti
- Rezultat je logička vrednost (boolean)

Operator	Meaning	Example	Outcome if age is 42
<	Less than	age < 21	false
<=	Less than or equal to	age <= 18	false
>	Greater than	age > 16	true
>=	Greater than or equal to	age >= 30	true

Logički operatori

- Negacija **!**
- Logičko „i“ (konjukcija) **&**
- Logičko „ili“ (disjunkcija) **|**
- Ekskluzivno „ili” **^**

- Skraćeno izračunavanje
  - Konjukcija je netačna ako je prvi operand netačan
  - Disjunkcija je tačna ako je prvi operand tačan
  - Tada nema potrebe proveravati vrednost drugog operanda
  - Uobičajeno se koriste skraćene varijante čime se izbegavaju nepotrebna izračunavanja

- Skraćena konjukcija **&&**
- Skraćena disjunkcija **||**

Operatori nad bitovima

- Negacija ~
- Konjukcija &
- Disjunkcija |
- Ekskluzivno ili ^
- Pomeranje bitova ulevo <<
- Pomeranje bitova udesno >>

Ostali operatori

- Ternarni operator za uslov
  - U zavisnosti od istinitosti prvog operanda (logički izraz) vraća vrednost drugog ili trećeg operanda
  - `x ? y : z`
- Podrazumevana vrednost tipa
  - `default` (`0` za vrednosne tipove, `null` za adresne)

if - else

- if – else

```
if (domaci > gosti)
  ishod = "1";
else if (domaci < gosti)
   ishod = "2";
else
   ishod = "x";
```

switch [1]

- Zavisno od vrednosti izraza u `switch` naredbi, izvršava se deo koda u okviru odgovarajuće `case` naredbe
- ako vrednost ne odgovara nijednoj `case` naredbi, izvršava se kod definisan u okviru labele `default` (ako postoji)
- Nema automatskog prelaska u narednu `case` naredbu
- svaki case se mora završiti naredbom skoka – `break` (najčešće) ili `goto`

switch [2]

```
switch (plasman)
{
   case 1:
       medalja = "zlato";
       break;
    case 2:
       medalja = "srebro";
       break;
    case 3:
       medalja = "bronza";
       break;
    default:
       medalja = "bez medalje";
       break;
}
```

Ciklusi - for

```
for (int i = 0; i < tekst.Length; i++)
{
    Console.WriteLine(tekst[i]);
}
```

Ciklusi – while, do - while

```
while (broj > 20)
{
   Console.WriteLine(broj);
    broj /= 2;
}

do
{
   Console.WriteLine(broj);
    broj /= 2;
}
while (broj > 20);
```

- Razlika – **`while`** može da se ne izvrši, **`do-while`** se izvršava bar jednom

Ciklusi – break, continue

- **`break`** – izlazak iz tekućeg ciklusa
- **`continue`** – prekid tekuće iteracije i prelazak na sledeću iteraciju

```
for (int i = 0; i < tekst.Length; ++i)
{
   if (tekst[i] == ' ')
       break;
   else if (tekst[i] != 'a')
       continue;
   brojSlovaA++;
}
```

Ciklusi – foreach

- Iteriranje kroz elemente kolekcije
  - **foreach (<tip> <identifikator> in <kolekcija>)**

```
foreach (char c in tekst)
{
  Console.WriteLine(c);
}
```

Komentari

- Jednolinijski `//tekst`
- Višelinijski `/* tekst */`
- Mogućnost automatskog generisanja dokumentacije na osnovu specijalnih komentara
- Jednolinijski dokumentacijski komentar `///tekst`
- Višelinijski dokumentacijski komentar `/** tekst */`
- Tagovi za dokumentacijske komentare
  - **`<remarks>`, `<summary>`, `<example>`, `<exception>`, `<param>`, `<permission>`, `<returns>`, `<seealso>`, `<include>`**