**ÚVOD DO PROGRAMOVANIA**

Program je sled príkazov. Každý program sa dá rozdeliť na menšie elementárne úlohy, ktoré sú riešiteľné príkazmi konkrétneho jazyka. Príkazy vykonáva procesor, údaje/dáta sú uložené v operačnej pamäti RAM.

Procesor teda vykonáva jednoduché príkazy (inštrukcie – RISC/CISC). Ich zápis sa nazýva strojový kód a je pre ľudí nečitateľný. Čiastočne čitateľnejšia forma je zápis kódu v jazyku assembler, každý procesor má však inú syntax tohto jazyka.

Čitateľnosť zlepšil príchod vyšších programovacích jazykov (napr. C). Tie sú písané príkazmi ako napr. while (pokiaľ), break (preruš) a pod. Takýto jazyk patrí do skupiny tzv. štruktúrovaných. Pre ešte ľahšie/názornejšie programovanie vznikli objektovo orientované jazyky (napr. C++/C#/Java), keďže všetko vôkol nás sa dá vnímať ako objekty s vlastnosťami a činnosťami, ktoré sa s nimi dajú robiť.

Kód programu písaný v prog. jazyku sa volá zdrojový kód. Tento kód spracuje program zvaný kompilátor, ktorý ho pretransformuje do strojového kódu konkrétnej platformy. Zdrojový kód môže byť vstupom aj pre interpréter, čo je program, ktorý priamo spúšťa program v sebe. Niečo ako emulátor. Ponúka multiplatformovosť, no zaťažuje HW oproti natívnemu programu.

Vývojové prostredie je program, ktorý slúži na pohodlné písanie kódu. Okrem editora môže obsahovať kompilátor, simulátor, opravovač/dopisovač kódu, modul na komunikáciu a pod.

Programy vytvárame v jazyku C#, ako konzolové aplikácie .NET Framework vo Visual Studio IDE. Konverziu a spustenie robíme cez klávesovú skratku CTRL+F5 (pozdrží zatvorenie okna).

**JAZYK C# - PREMENNÁ**

Základným prvkom každého programu je premenná, keďže program pracuje s dátami. Programátor nevie, aké dáta budú, preto ich univerzálne v kóde reprezentuje premennou (odkaz na časť pamäte, kde sa dáta nachádzajú).

Typy premenných:

*int cislo = 1; // cele cislo*

*float desatinne = 3.14f; // zakladne desatinne cislo*

*double dvojnasobne = 3.14; // presnejsie desatinne cislo*

*char znak = ‘1‘; // znak sa pise do apostrofu*



*string text = “1“; // pouzivaju sa uvodzovky*

*bool bolean = true; // resp. false*

*var univerzalnyTyp; // typ sa prisposoby podla toho, aka hodnota sa don PRVA zapise*

-deklarácia / inicializácia / definícia

-lokálne / globálne

-existencia v rámci blokov a prekrytie

-pravidlá pre tvorbu názvov (znaky / čísla / ťaví zápis / prvé písmeno )

-modifikátory (unsigned, signed, short, long, const)

Príklad:

static void Main(string[] args)

{

int c1; // vytvorenie premennej pre cele cislo

int c2 = 0; // vytvorenie a naplnenie nulou

Console.WriteLine("Zadaj číslo 1:"); // vypis textu

c1 = int.Parse( Console.ReadLine() ); // nacitanie a konvertovanie



Console.WriteLine("Zadaj číslo 2:"); /\* koment od-do \*/

string cislo2 = Console.ReadLine(); // nacitanie textu

c2 = Convert.ToInt32(cislo2); // konverzia stringu na integer

Console.WriteLine("Súčet: " + (c1 + c2) ); // automaticka konv. i->s

Console.WriteLine("Rozdiel: " + (c1 - c2).ToString() ); // cez metodu

Console.WriteLine("Podiel: " + Convert.ToString(c1 / c2) ); // cez inu

int sucin = c1 \* c2; // zapis vypoctu do pomocnej premennej

Console.WriteLine("Súčet: " + sucin ); // vypis pomocnej premennej

}

*Úloha: Vytvorte program, ktorému zadáte rok narodenia a aktuálny rok a on vypíše, koľko máte rokov.*

**JAZYK C# - OPERÁTORY**

Operátory slúžia na prácu s premennými (aritmetické, binárne / logické, relačné):

|  |  |  |
| --- | --- | --- |
| **SYNTAX** | **VÝZNAM** | **POUŽITIE** |
| + | aritmetický súčet (sčítanie) | 5 + 2 (7) |
| - | aritmetický rozdiel (odčítanie) | 5 – 2 (3) |
| \* | aritmetický súčin (násobenie) | 5 \* 2 (10) |
| / | aritmetický podiel (delenie) | 5 / 2 (2) |
| % | zvyšok po celočíselnom delení (modulo) | 5 % 2 (1) |
| ++ | inkremenácia (zvýšenie o 1) | 5++ (6) |
| -- | dekrementácia (zníženie o 1) | 5-- (4) |
| = | priradenie | a = 5 |



|  |  |  |
| --- | --- | --- |
| **SYNTAX** | **VÝZNAM** | **POUŽITIE** |
| & | bitový súčin (AND) | 1001 & 1010 (1000) |
| | | bitový súčet (OR) | 1001 | 1010 (1011) |
| ^ | bitové vylúčenie (XOR) | 1001 ^ 1010 (0011) |
| ~ | bitová negácia (komplement) | ~1001 (0110) |
| << | bitový posun vľavo (shift vľavo) | 01 << 2 (0100) |
| >> | bitový posun vpravo (shift vpravo) | 0110 >> 2 (01) |
| && | logický súčin (AND) | POD1 && POD2 |
| || | logický súčet (OR) | POD1 || POD2 |
| ! | logická negácia (NOT) | !true (false) |
| ? | podmienkový operátor | POD ? PR1 : PR2 |

|  |  |  |
| --- | --- | --- |
| **SYNTAX** | **VÝZNAM** | **POUŽITIE** |
| == | test rovnosti | A == B |
| != | test nerovnosti | A != B |
| < | test menšieho | A < B |
| <= | test menšieho alebo rovného | A <= B |
| > | test väčšieho | A > B |
| >= | test väčšieho alebo rovného | A >= B |

Pre generovanie náhodných čísel treba vytvoriť objekt, z ktorého čísla budeme generovať. Metóda na generovanie sa volá Next() a má tri druhy preťaženia.

Random rrr = new Random(); // vytvorenie objektu pre generovanie

int nahoda = rrr.Next(1, 7); // vygenerovanie cisla od 1 do 6

*Úloha: Vytvorte program, ktorý vypíše 3 mená a k ním náhodne tri čísla známok (1-5)*

**JAZYK C# - VETVENIE**

Vetvenie sa používa vtedy, pokiaľ chceme, aby sa určitá časť kódu vykonala za určitých podmienok a za iných podmienok sa vykonala nevykonalo nič alebo niečo iné.

if() – jednoduché vetvenie, pokiaľ platí podmienka, kód sa vykoná, inak sa nevykoná

if() – kód v bloku za if sa vykoná pokiaľ platí podmienka, inak sa vykoná kód v bloku za else  
else

if() – postupne sa testujú podmienky, pokiaľ niektorá platí, vykoná sa daný blok, zvyšok sa preskočí  
else if()  
else if()  
else

switch() – vetvenie, ktoré testuje rovnosť dvoch hodnôt. Používa sa pri výberoch, stavoch a pod.

Príkazy if a if-else if-else if-else nie sú samostatné príkazy, ide o modifikáciu vetvenia if()-else. Platí totiž, že časť else sa nemusí písať.

Pri vetvení aj cykloch platí, že pokiaľ je v bloku jeden príkaz, zátvorky sa môžu a nemusia písať.

Príklad:

int cislo1; // premenné pre dve čísla a text

int cislo2;

string operacia;

Console.WriteLine("Zadaj číslo 1: "); // výpis na obrazovku

cislo1 = int.Parse( Console.ReadLine() ); // načítanie INT

Console.WriteLine("Zadaj číslo 2: ");

cislo2 = Convert.ToInt32( Console.ReadLine() );

Console.WriteLine("Zadaj operáciu (+-\*/): ");

operacia = Console.ReadLine(); // načíta sa STRING a NEkonvertuje sa

if(operacia == "+") // ak sa zadal znak +, tak ...

Console.WriteLine("Súčet bol: " + (cislo1+cislo2).ToString() );

else if (operacia == "-") // ak sa zadal znak -, tak ...

Console.WriteLine("Rozdiel bol: " + (cislo1 - cislo2).ToString());

else if (operacia == "\*")

Console.WriteLine("Súčin bol: " + (cislo1 \* cislo2).ToString());

else if (operacia == "/")

Console.WriteLine("Podiel bol: " + (cislo1 / (float)cislo2 ).ToString());

else

Console.WriteLine("ERROR. Neznáma operácia!");

Príklad:

Random r = new Random(); // vytvorenie objektu pre náhodné generovanie

int znamka = r.Next(1, 6); // vygenerovanie čísla medzi 1 a 5

string slovne; // pomocná premenná pre text

switch(znamka)

{

case 1: // ak sa znamka == 1 ,tak...

slovne = "výborný"; // zapíš text do premennej

break; // ukonči vetvenie switch

case 2: // ak sa znamka == 2 ,tak...

slovne = "chválitebný";

break;

case 3: // ak sa znamka == 3 ,tak...

slovne = "dobrý";

break;

case 4:

slovne = "dostatočný";

break;

case 5:

slovne = "nedostatočný";

break;

default: // ak sa znamka nerovnala 1,2,3,4 alebo 5 ,tak...

slovne = "ERROR";

break;

}

Console.WriteLine("Monika bola hodnotená: " + slovne); // výpis

**JAZYK C# - CYKLUS**

Pokiaľ sa má vykonať nejaký kód viackrát, je možné použiť príkaz cyklu.

for( int i=0; i<XXX; i++) { } // cyklus sa zopakuje XXX\_krát a „i“ sa bude zvyšovať od 0 po XXX-1

while(i<XXX) { } // cyklus bude prebiehať, pokým bude platiť podmienka (i < XXX)

do { } while(i<XXX); // najskôr sa vykoná kód za „do“, potom sa až otestuje, či sa to má zopakovať

for( var i in KOLEKCIA) { } // cyklus prejde toľkokrát, koľko je prvkov v kolekcii a „i“ rep. daný prvok

Príklad:

Random r = new Random(); // objekt pre náhodné generovanie

int jednotiek = 0; // pomocné premenné pre rátanie čísel

int dvojok = 0;

int trojok = 0;

for (int i = 0; i < 1000; i++) //cyklus opakujúci sa 1000x

{

int tmp = r.Next(1, 4); // generovanie čísla 1-3

if (tmp == 1) // ak sa vygenerovalo číslo 1...

jednotiek++; // ...zvýš počet jednotiek

else if (tmp == 2) // ak sa vygenerovalo číslo 2...

dvojok++; // ...zvýš počet dvojok

else // ak sa vygenerovalo číslo 3...

trojok++; // ...zvýš počet trojok

}

Console.WriteLine("Jednotiek bolo: " + jednotiek.ToString() );

Console.WriteLine("Dvojok bolo: " + dvojok.ToString()); // výpis

Console.WriteLine("Trojok bolo: " + trojok.ToString());

Príklad:

Random r = new Random(); // objekt pre náhodné generovanie

int poslednNahoda = 0; // premenná pre posledné vygen. číslo

for (int i = 0; i < 10; i++) // cyklus opakujúci sa 10x

{

int nahoda = r.Next(1, 4); // vyygenerovanie čísla 1-3

while(poslednNahoda == nahoda) // pokiaľ bude nové rovnaké...

{

nahoda = r.Next(1, 4); // ...PREgeneruj nové číslo 1-3

}

poslednNahoda = nahoda; // zapamätanie si posledného vygen. čísla

Console.WriteLine(nahoda); // výpis náhodného (aktual==predch) čísla

}

*Úloha: Superman vs Batman (odčítavanie od 100 HP silu úderu (0-9), pokým niekto žije)*

**JAZYK C# - POLIA, LISTY**

Pokiaľ potrebujeme uložiť viacero dát pod jedným názvom (napr. mať 1000 čísel a nemať pre každé z nich vlastnú premennú), je vhodné využiť pole. Všetky prvky v ňom môžu byť len jedného typu.

int[] poleCisel = new int[1000]; // pole pre 1000 celých čísel

Pokiaľ je v poli iné pole, hovoríme o viacrozmernom poli. Dvojrozmerné sa dá vnímať ako tabuľka (prvý rozmer sú riadky, druhý stĺpce. Počet a typ hodnôt v „podpoliach“ je rovnaký.

int[,] dvojrozmernePole = new int[10,2]; // pole obsahujúce 10 polí. Každé ma 2 int

Príklad:

Random rnd = new Random(); // objekt pre nahodne cisla

int[] vyskaTrpaslikov = new int[7]; // pole pre 7 INT

for (int i = 0; i < vyskaTrpaslikov.Length; i++) // cyklus prejde 7x

{

vyskaTrpaslikov[i] = rnd.Next(50,100); // kazdemu prvku sa priradi RND

}

int najvacsi = vyskaTrpaslikov[0]; // prehlasenie, ze najvacsi je prvy

for (int i = 1; i < vyskaTrpaslikov.Length; i++) // cyklus prejde 6x (od 1)

{

if(najvacsi < vyskaTrpaslikov[i]) // ak DALSI je vacsi...

najvacsi = vyskaTrpaslikov[i]; // ...tak si uloz novu vysku

}

for (int i = 0; i < vyskaTrpaslikov.Length; i++) // cyklus prejde 7x

Console.WriteLine(i.ToString() + ". trpaslik: " + vyskaTrpaslikov[i]);

Console.WriteLine("Najvacsi trpasli ma velkost: " + najvacsi.ToString());

*Úloha: z 10000 náhodných čísel od 0-99, zistiť, koľko bolo 0, 1, 2, 3, 4... vygenerovaných*

*Úloha: vytvorte pomocou 2D poľa výpis rozvrhu do tvaru tabuľky*

List ponúka na rozdiel od poľa hlavne výhodu v tom, že jednotlivé prvky sa môžu priebežne pridávať, odoberať, meniť ... Práca s ním je dynamickejšia, obsahuje množstvo metód. Stále platí, že sa doň zapisuje len jeden typ dát. Aby bolo možné použiť list, je nutné prilinkovať priestor Generic:

using System.Collections.Generic;

Foreach je cyklus, ktorý postupne vytvorí kópiu (item) každého prvku z kolekcie. Oproti for() nevieme priamo získať index iterácie a nemôžeme meniť prvok priamo. Je však prehľadnejší. Zápis:

foreach (var item in zoznamMien)

Príklad:

List<string> zoznamMien = new List<string>(); // list textov (zatial prazdny)

while (true) // zacykleny cyklus "Donekonecna a este dalej"

{

Console.WriteLine("Počet zadaných mien: " + zoznamMien.Count);

Console.WriteLine("Zadaj NOVÉ meno:"); // vypis do konzoly

string tmp = Console.ReadLine(); // nacitanie textu z konzoly

// 1. MOZNOST

//bool jeVZozname = zoznamMien.Contains(tmp); // existujuca metoda...

// 2. MOZNOST

bool jeVZozname = false;

foreach (var item in zoznamMien)

if (item == tmp)

jeVZozname = true;

if (jeVZozname == false)

zoznamMien.Add(tmp);

else

Console.WriteLine("Meno je už v zozname!!!");

}

*Úloha: vytvorte program, v ktorom bude môcť užívateľ zadávať, vypisovať, prepisovať a vymazávať položky v liste, napr. menný zoznam účastníkov krúžku*

**JAZYK C# - METÓDY**

Pokiaľ sa nejaký kód volá viackrát na rôznych miestach programu, je možné ho napísať len raz a v daných miestach ho len zavolať. Takýto „podprogram“ sa nazýva funkcia / metóda / procedúra. V programoch sme ju už veľakrát používali (Next(), WriteLine(), ToInt32(), ToString() ... ).

Vytvorenie vlastnej sa robí tak, že mimo inú metódu sa uvedie návratový typ (pokiaľ metóda nemá nič navrátiť, použije sa identifikátor “void“), následne názov a nakoniec do zátvoriek vstupné parametre (pokiaľ žiadne nie sú, zátvorky budú prázdne).

Príkaz “return“ okamžite ukončuje funkciu. Môže sa použiť aj v metóde, ktorá má návratový typ void skoršie ukončenie. Tvar: *return;*

Metódy sa požívajú nie len na zmenšenie kódu (lebo sa definujú len raz a volajú viackrát), ale aj na prehľadnosť (zadelenie súvisiaceho kódu do pomenovaného bloku a následne len volanie blokov).

*Pozn.: keďže v konzolovom C# projekte je main() typu STATIC, musí byť každá volaná funkcia tiež!*

Vzor metódy, ktorá navracia celé číslo, volá sa “metoda“ a pri volaní očakáva dve celé čísla:

*int metoda(int a, int b)   
{  
 return a+b;  
}*

Na rozdiel od premennej, názov metód sa môže opakovať. Aby však program vedel, ktorú z nich použiť, musia sa líšiť typom alebo počtom vstupných parametrov. Ide o tzv. pretypovanie.

Príklad:

static void Main(string[] args)

{

int vysledok = kalkulacka("+", 10, 2);

Console.WriteLine(vysledok);

Console.WriteLine(kalkulacka("-", 10.2f, 5f));

}

static int kalkulacka(string operacia, int cislo1, int cislo2)

{

if (operacia == "+")

return cislo1 + cislo2;

else if (operacia == "-")

return cislo1 - cislo2;

else if (operacia == "\*")

return cislo1 \* cislo2;

else if (operacia == "/")

return cislo1 / cislo2;

else return 0;

}

static float kalkulacka(string operacia, float cislo1, float cislo2)

{

if (operacia == "+")

return cislo1 + cislo2;

else if (operacia == "-")

return cislo1 - cislo2;

else if (operacia == "\*")

return cislo1 \* cislo2;

else if (operacia == "/")

return cislo1 / cislo2;

else

return 0;

}

*Úloha: vytvorte metódu, ktorá zobrazí do tvaru tabuľky 2D pole (rozvrh). Zavolajte ju po stlačení klávesu z.*

static Random r = new Random(); // globálne, aby sa generovalo nové vždy iné

static void Main(string[] args)

{

int hrac1 = 0;

int hrac2 = 0;

while (hrac1 < 100 && hrac2 < 100)

{

hrac1 += hodKockou();

hrac2 += hodKockou();

Console.WriteLine("Hrac 1 ma: " + hrac1.ToString());

Console.WriteLine("Hrac 2 ma: " + hrac2.ToString());

Console.WriteLine("---------------");

}

if (hrac1 >= 100)

Console.WriteLine("Vyhral hrac 1!");

else

Console.WriteLine("Vyhral hrac 2!");

}

static int hodKockou()

{

int hod = r.Next(1, 7);

int sucet = hod;

while (hod == 6)

{

hod = r.Next(1, 7);

sucet += hod;

}

return sucet;

}

**JAZYK C# - PRÁCA SO SÚBOROM**

Jedným zo spôsobov, ako pracovať so súbormi, je využitie Streamu. Metódy a objekty pre prácu s ním sa nachádzajú v mennom priestore System.IO. Objekt pre prácu so súborom je typu StreamReader. Po dokončení práce so súborom sa má súbor/stream uzatvoriť cez Close().

Príklad:

/\* OBSAH SUBORU mena.txt

Adam

Boris

Cyril

David

\*/

StreamReader subor = new StreamReader("c:\\a\\mena.txt");

while (true)

{

string obsah = subor.ReadLine(); // nacitanie celeho riadku

if (obsah == null) // test, ci nie je koniec suboru

break; // ukoncenie zacykleneho nekonecneho cyklu

Console.WriteLine(obsah); // vypis nacitaneho riadku

}

subor.Close();

Príklad:

/\* OBSAH EXCEL SUBORU zosit.csv

Adam;5

Boris;1

Cyril;3

David;1

\*/

StreamReader subor = new StreamReader("c:\\a\\zosit.csv");

List<string> mena = new List<string>();

List<int> znamky = new List<int>();

while (true)

{

string obsah = subor.ReadLine();

if (obsah == null)

break;

// rozdelenie riadku podla bodkociarok na viac textov

string[] poleStringov1Riadku = obsah.Split(';');

mena.Add( poleStringov1Riadku[0] ); // prvy text je meno

znamky.Add( int.Parse( poleStringov1Riadku[1] ) ); // druhy znamka

}

Console.WriteLine("MENA");

foreach (var item in mena)

Console.WriteLine(item);

Console.WriteLine("ZNAMKY");

foreach (var item in znamky)

Console.WriteLine(item);

subor.Close();

*Úloha: vytvorte program, ktorý načíta zo súboru mená, známky a triedy a uloží ich do samostatných súborov.*

*Úloha: vytvorte program, ktorý načíta zo súboru mená, známky a triedy a vypíše priemernú známku každej triedy (sčíta známky v rámci rovnakých tried) a vypíše ich v poradí od najlepšieho priemeru.*