**O’ZBEKISTON RESPUBLIKASI MUDOFAA VAZIRLIGI AXBOROT-KOMMUNIKATSIYA TEXNOLOGIYALARI VA ALOQA HARBIY INSTITUTI**



**Axborot texnologiyalari va dasturiy injinering kafedrasi**

**Obyektga yo`naltirilgan dasturlash fanidan**

Mustaqil ishi

**Mavzu: Ma`lumot turlarini biridan ikkinchisiga o`zgartirish**

Bajardi: **146-19** guruh k-nt **Mamajonov D.**

Qabul qildi: l-nt Norbekov A.

**Toshkent 2022**

Reja:

1. C# dasturlash tili haqida dastlabki ma’lumotlar va uning yaratilish tarixi.
2. C# tilining asosiy xususiyatlari.
3. C# dasturlash tilining tashkil etuvchilari, til alfaviti va leksemasi.
4. Ma`lumot turlarini biridan ikkinchisiga o`zgartirish.

**C# dasturlash tili haqida dastlabki ma’lumotlar va uning yaratilish**

**tarixi.**

C# tili 2000 yilda Microsoft kompaniyasining Anders Xeylsberg (Anders

Hejlsberg) boshchiligidagi ishchi guruhi tomonidan yaratilgan. Andersom

Xeylsberg IBM oilasiga masub kompyuterlar uchun birinchi kompilyatsiya qiluvchi

dasturlash tillaridan biri Turbo Pascal ni yaratgan dasturchi sifatida ko‘pchilikga

tanishdir.

C# dasturlash tili 2000 yilda ishlab chiqilgan, lekin uning rasmiy 1.0 versiyasi

2002 yilda ishlab chiqildi. Bunga sabab aynan shu yili Windowsning .NET

platformasi 1.0 versiyasi ishlab chiqilganligidir.

.NET FrameWork Windows operatsion tizimi(OT) muhitida ishlovchi barcha

dasturlash tillari uchun ochiq bo‘lgan va OT ning barcha komponentalariga

dasturlash tili orqali murojaat qilishni ta’minlovchi dasturlash platformasi

hisoblanadi. .NET FrameWork barcha dasturlash tillari uchun umumiy bo‘lgan

bazaviy sinflar va ularni bajarilishini ta’minlovchi Common Language

Runtime(CLR) interpretatoridan tashkil topgan. Bunda dasturlash tili tomonidan

kompilyatsiya qilingan har qanday dastur ishga tushirilganda CLR tomonidan

baytkodlarga o‘zgartiriladi va xuddi Assembler singari OT da ishlash xolatiga

o‘tkaziladi, ya’ni interpretatsiya qilinadi. To‘g‘ri bu jarayon dasturning ishga

tushishi va yuklanish jarayonini biroz sekinlashtiradi. Lekin hozirda yaratilayotgan

kompyuter texnikalari uchun bu muammo emas.

C# dasturlash tili Windows ning .NET platformasi bilan birgalikda ishlashga

mo‘ljallangan deyarli 100% ob’ektga mo‘ljallangan dasturlash tili hisoblanadi.

**C# tilining asosiy xususiyatlari:**

• NET Framework sinflarini to‘liq 100% ishlatish, obyektli dasturlash va

sinflarni to‘la qo‘llash, avlod qoldirish, inkapsulyatsiya, polimorfizm va

virtualizatsiya metodlarini qo‘llash, operatorlarni qayta yuklash va virtual

funksiyalarni yaratish va foydalanish;

• Asosiy va qo‘shimcha tiplarning to‘la to‘plami va yaratish imkoniyati;

• integratsiyalashgan XML-xujjatlarni avtomatik generatsiya qilish imkoniyati;

• Dinamik taqsimlangan xotirani avtomatik tozalash.

• Sinflar va metodlarni alohida atributlar bilan belgilab qo‘yish imkoniyati, ya’ni

ayrim metod va sinflar faqatgina tekshirish(otladka) rejimida kompilyatsiya

qilinadi.

• Windows API ga oson murojaat qilish imkoniyati;

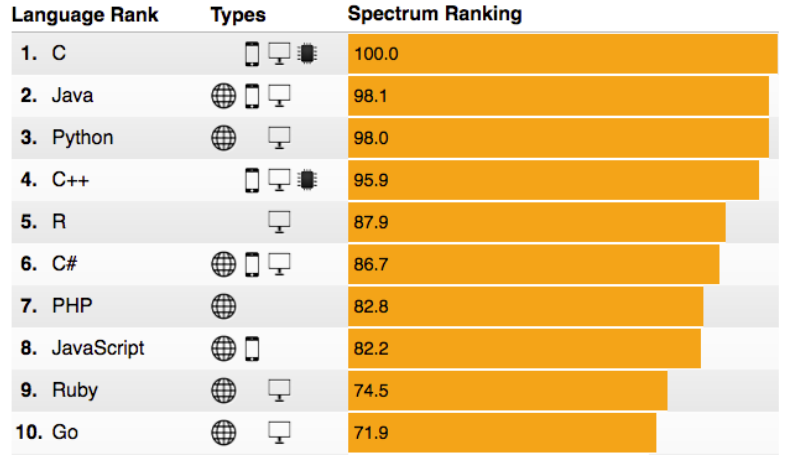
• Xotiraga va ko‘rsatkichlarga zarurat tug‘ilganda to‘g‘ridan-to‘g‘ri murojaat

qilish imkoniyati;

• Xodisalarni qo‘llab quvvatlash;

• Dinamik WEB - sahiflarni yaratish(ASP.NET) va h.

IEEE Spectrum reytingi:



**C# dasturlash tilining tashkil etuvchilari, til alfaviti va leksemasi.**

Dasturlash tilini o‘rganishni biror bir xorijiy tilni o‘rganishga qiyoslash

mumkin. Xorijiy tilni o‘rganish odatda tilning alfaviti o‘rganishdan boshlanadi,

so‘ngra so‘zlarni, ularning tarjimasini o‘rganishga o‘tiladi, keyin esa ana shu

so‘zlardan foydalanib jumlalar, gap tuzishga o‘tiladi va sekin astalik bilan so‘z

boyligini ko‘paytirib borgan holda hamda ko‘p mashq qilish, ko‘p bora muloqot

qilish natijasida ushbu tilda o‘z fikrlarini erkin ifoda etishga, tilni mukammal

darajada bilishga erishiladi. Shunga o‘xshash biror dasturlash tilini o‘rganishda

mana shu ketma-ketlikdan foydalanish yaxshi samara beradi.

Endi C# dasturlash tilining alfaviti bilan tanishsak. C# dasturlash tilida

matnlarni yozishda Unicode kodirovkasidagi belgilardan foydalaniladi. Kodirovka

– bu belgilar va unga mos keluvchi sonli kodlar jadvalidir. Unicode kodirovkasi bir

vaqtning o‘zida barcha alfavitlardagi belgilardan foydalanish imkonini beradi.

C# alfaviti quyidagilarni o‘z ichiga oladi:

• lotin alfavitlar hamda ular bilan birgalikda qo‘llash uchun ostki chiziq(\_);

• raqamlar;

• maxsus belgilar, masalan +, \*, { i & ;

• bo‘sh va tabulyatsiya belgilari;

• qatorni ko‘chirish belgilari.

Belgilar to‘plamidan foydalanib, leksemalar va izohlarni yoziladi.

Leksema (token) — bu dasturlash tilining mustaqil holda biror ma’noni

bildiruvchi eng kichik birligidir. Leksemening quyidagi ko‘rinishlari mavjud:

• nom yoki ism (identifikatorlar);

• kalit so‘zlar;

• amal belgilari;

• ajratgichlar;

• literallar (konstantalar).

Leksemaga misol sifatida quyidagilarni ko‘rsatishimiz mumkin: Vasia nomli

identifikator, goto kalit so‘zi,+ amal belgisi va sh.k.

Leksemalardan foydalangan holda operatorlar va ifodalar tashkil qilinadi.

Masalan: a+b ifoda, u ikki kattalikni qo‘shishni bildiradi;

int a; - bu a nomli o‘zgaruvchini e’lon qilish operatoridir.

**Ma`lumot turlarini biridan ikkinchisiga o`zgartirish.**

Ma'lumotlar turlarini ko'rib chiqishda ma'lum bir turdagi qanday qiymatlarga ega bo'lishi va qancha bayt xotirani egallashi ko'rsatilgan. Oxirgi mavzuda arifmetik amallar ko'rib chiqildi. Endi har xil turdagi ma'lumotlarga qo'shish amalini qo'llaymiz:

|  |  |
| --- | --- |
| 1  2 | byte a = 4;  int b = a + 70; |

Operatsiya natijasi, kutilganidek, 74 raqamidir.

Ammo endi bayt tipidagi ikkita obyektga qo'shishni qo'llashga harakat qilaylik :

|  |  |
| --- | --- |
| 1  2 | byte a = 4;  byte b = a + 70; // xato |

Bu yerda faqat qo'shish natijasini oladigan o'zgaruvchining turi o'zgargan - int dan baytga. Biroq, dasturni kompilyatsiya qilinganda, kompilator xatolik haqida xabar beradi. Va agar Visual Studio'da ishlayonayotgan bo'lsa, muhit ikkinchi qatorni qizil chiziq bilan chizib, unda xatolik borligini ko'rsatadi. Ishlayotganda, ma'lum bir turdagi saqlashi mumkin bo'lgan qiymatlar oralig'ini hisobga olish kerak. Ammo bu holda, olinishi kutilayotgan 74 raqami bayt tipidagi qiymatlar oralig'ida, shunga qaramay, xatolik mavjud. Chunki, qo‘shish (va ayirish) operatsiyasi, agar operatsiya int dan kichik yoki unga teng bo‘lgan butun sonli ma’lumotlar turlarini (ya’ni bayt, short, int turlarini) o‘z ichiga olsa, int tipidagi qiymatni qaytaradi. Shuning uchun operatsiya natijasi a + 70 xotirada uzunligi 4 bayt bo'lgan obyekt bo'ladi. Keyin bu obyektni bayt tipiga ega bo'lgan va xotirada 1 baytni egallagan b o'zgaruvchisiga belgilashga harakat qilamiz.

Va bu vaziyatdan chiqish uchun siz turdagi konvertatsiya operatsiyasini qo'llashingiz kerak. Turni o'zgartirish operatsiyasi qavslar ichida qiymat o'zgartirilishi kerak bo'lgan turni ko'rsatishni o'z ichiga oladi:

Shunday qilib, turni o'zgartirish operatsiyasini qo'llash orqali oldingi misolni o'zgartiramiz:

|  |  |
| --- | --- |
| 1  2 | byte a = 4;  byte b = (byte)(a + 70); |

* 1. **Toraytiruvchi va kengaytiruvchi transformatsiyalar**

Transformatsiyalar torayuvchi (torayuvchi) va kengaytiruvchi (kengayuvchi) bo'lishi mumkin. Kengaytirilgan konversiyalar xotiradagi ob'ekt hajmini kengaytiradi. Misol uchun:

|  |  |
| --- | --- |
| bitta  2 | byte a = 4; // 0000100  ushort b = a; // 000000000000100 |

Bunda ushort tipidagi o'zgaruvchiga bayt tipidagi qiymat beriladi. Bayt turi 1 baytni (8 bit) egallaydi va a o'zgaruvchining ikkilik qiymati quyidagicha ifodalanishi mumkin:

|  |  |
| --- | --- |
| 1 | 00000100 |

Qisqa qiymat 2 bayt (16 bit). Va b o'zgaruvchini tayinlashda a o'zgaruvchining qiymati 2 baytgacha kengaytiriladi

|  |  |
| --- | --- |
| 1 | 0000000000000100 |

Ya'ni, 8 bitni egallagan qiymat 16 bitgacha kengaytiriladi .

Toraytiruvchi konvertatsiyalar esa, qiymatni kamroq bitlik turiga

toraytiradi. Maqolaning ikkinchi ro'yxatida biz faqat toraytirilgan o'zgarishlar bilan shug'ullandik:

|  |  |
| --- | --- |
| 1  2 | ushort a = 4;  byte b = (byte) a; |

Bu erda 8 bitni egallagan b o'zgaruvchisiga 16 bitni egallagan ushort qiymati beriladi. Ya'ni, 0000000000000100 bizdan olamiz 00000100. Shunday qilib, qiymat 16 bitdan (2 bayt) 8 bitgacha (1 bayt) torayadi.

* 1. **Aniq va yashirin konvertatsiyalar**

Yashirin konvertatsiyalar

Konversiyalarni kengaytirganda, kompilyator biz uchun barcha ma'lumotlarni o'zgartirishni amalga oshirdi, ya'ni konvertatsiyalar yashirin konvertatsiyalar edi

. Bunday o'zgarishlar hech qanday qiyinchilik tug'dirmaydi. Shunga qaramay, bunday o'zgarishlarning umumiy mexanikasi haqida bir necha so'z aytish kerak. Agar konversiya kichikroq bit chuqurligining belgisiz turidan kattaroq bit kengligining belgisiz turiga o'tkazilsa, u holda qiymatlari 0 bo'lgan qo'shimcha bitlar qo'shiladi. Bu nol to'ldirish yoki nol kengaytma deyiladi.

|  |  |
| --- | --- |
| 1  2 | byte a = 4; // 0000100  ushort b = a; // 000000000000100 |

Agar belgilangan turga o'tkazish amalga oshirilsa, agar raqam musbat bo'lsa, bit tasviri nol bilan to'ldiriladi, agar raqam manfiy bo'lsa. Raqamning oxirgi bitida ishora biti mavjud - musbat raqamlar uchun 0 va manfiy raqamlar uchun

1. Qo'shilgan bitlarga kengaytirilganda, belgi biti kompilyatsiya qilinadi. Musbat raqamni aylantirishni o'ylab ko'ring:

|  |  |
| --- | --- |
| 1  2 | sbyte a = 4; // 0000100  short b = a; // 000000000000100 |

Manfiy raqamni aylantirish:

|  |  |
| --- | --- |
| 1  2 | sbyte a = -4; // 1111100  short b = a; // 111111111111100 |

Aniq konvertatsiyalar

Aniq konvertatsiyalar ( aniq konvertatsiya ) bilan biz o'zimiz konvertatsiya operatsiyasini (operatsiyani ()) qo'llashimiz kerak. Turni o'zgartirish operatsiyasining mohiyati shundan iboratki, qiymatdan oldin qavs ichida berilgan qiymat qaysi turdagi bo'lishi kerakligi ko'rsatilgan:

|  |  |
| --- | --- |
| 1  2  3 | int a = 4; int b = 6;  byte c = (byte)(a+b); |

Kompilyator kichikroq bit kengligi bo'lgan turdan kattaroq bit kengligiga ega bo'lgan turga kengaytiruvchi konvertatsiyalarni amalga oshiradi. Bu quyidagi transformatsiyalar zanjiri bo'lishi mumkin:

bayt -> short -> int -> long -> decimal int -> double

short -> float -> double char -> int

Barcha xavfsiz avtomatik konvertatsiyalarni quyidagi jadvalda tasvirlash mumkin:

|  |  |
| --- | --- |
| Turi | Qaysi turlarga xavfsiz tarzda aylantirilishi mumkin |
| bayt | short, ushort, int, uint, long, ulong, float, double, kasr |
| sbyte | short, int, long, float, double, decimal |
| short | int, long, float, double, decimal |
| ushort | int, uint, long, ulong, float, double, decimal |
| int | long, float, double, decimal |
| uint | long, ulong, float, double, decimal |

|  |  |
| --- | --- |
| long | float, double, decimal |
| ulong | float, double, decimal |
| float | Double |
| char | ushort, int, uint, long, ulong, float, double, decimal |

Aks holda, aniq turdagi konversiyalardan foydalanish kerak.

Shuni ham ta'kidlash kerakki, ikkilik va o'nlik kasr ma'lumotlarini saqlashi mumkin va o'nlik ikki barobardan kattaroq bit chuqurligiga ega bo'lishiga qaramay, siz baribir ikkilamchi qiymatni o'nlik kasr turiga aniq o'tkazishingiz kerak:

|  |  |
| --- | --- |
| 1  2 | double a = 4.0;  decimal b = (decimal)a; |

Ma'lumotlar yo'qolishi va tekshirilgan kalit so'z

Boshqa vaziyatni ko'rib chiqing, masalan, quyidagi holatda bo'ladi:

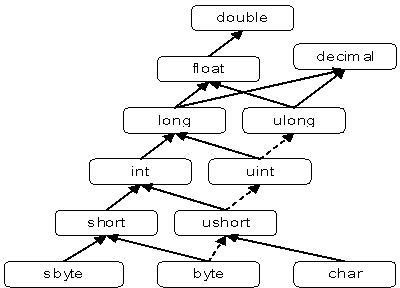
|  |  |
| --- | --- |
| 1  2  3  4 | int a = 33; int b = 600;  byte c = (byte)(a+b);  Console.WriteLine(c); // 121 |

Natijada 121 bo'ladi, shuning uchun 633 bayt turi uchun diapazondan tashqarida va yuqori bitlar kesiladi. Natijada 121 raqami bo'ladi. Shuning uchun konvertatsiya qilishda buni hisobga olish kerak. Va bu holda, biz jami 255 dan ko'p bo'lmagan raqamni beradigan a va b raqamlarini olishimiz mumkin yoki biz bayt o'rniga boshqa ma'lumot turini tanlashimiz mumkin, masalan, int. Biroq, vaziyatlar boshqacha bo'lishi mumkin. Biz a va b qiymatlari qanday bo'lishini aniq bilmasligimiz mumkin. Va bunday vaziyatlardan qochish uchun C# da kalit so'z mavjud checked:

|  |  |
| --- | --- |
|  | try  {  int a = 33; |

|  |  |
| --- | --- |
|  | int b = 600;  byte c = checked((byte)(a + b)); Console.WriteLine(c);  }  catch (OverflowException ex)  {  Console.WriteLine(ex.Message);  } |

Kalit so'zdan foydalanganda, checkeddastur to'lib-toshgan istisnoni chiqaradi. Shuning uchun, bu holda, try...catch konstruktsiyasi uni qayta ishlash uchun ishlatiladi. Biz ushbu konstruksiya va istisno holatlarini keyinroq batafsil ko‘rib chiqamiz, ammo hozircha biz bilishimiz kerakki, biz try blokida xatolik yuzaga kelishi mumkin bo‘lgan harakatlarni o‘z ichiga olamiz va xatoni catch blokida hal qilish mumkin.



Tiplarni oshkor bo‘lmagan arifmetik o‘zgartirish ketma-ketligi.