3 C# тілінде қауіпсіз емес бағдарламалау

3.1 Windows жүйесінде идентификаторларды белгілеу

Windows жүйесінің функцияларымен жұмыс істеуші барлық жүйелік бағдарламалаушылар жүйеде қабылданған (венгер жазбалары деп аталатын) идентификаторлардың белгілеуін түсінуге тиіс. Осы белгілерді Microsoft-тың ең жақсы бағдарламалаушыларының бірі Чарльз Симонии (Венгрияда туған) қолдана бастады. Қылжақтап осы белгілерді венгер жазбалары деп атала басталды және барлық ізбасарлар мен пайдаланушыларға (Windows-та бағдарламаларды әзірлейтіндерге) Windows-та қабылданған белгілердің ерекшеліктерімен танысу қажет болады. Идеяның мәні айнымалы, типтер және т.б. атаулары үшін олардың типін және бар ақпараттың сипатын суреттейтін префикстер қолданылатындығында. Кейбір префикстер келесі тізімде көрсетілген:

Префикс	Мәні
a	– Массив;
в	– Логикалық тип (BOOL);
by	– Таңбасы символдық тип (BYTE);
c	– Символдық тип;
cb	– Байттардың санауышы;
cr	– Түс;
cx,cy	– Қысқа тип (SHORT);
dw	– Таңбасы жоқ ұзын тип (DWORD);
fn	– Функция;
h	– Логикалық нөмер (HANDLE немесе HINSTANCE);
i	– Бүтін;
m_	– Клас айнымалысы;
n	– SHORT немесе int;
np	– жақын нұсқағыш;
p	–Нұсқағыш;
l	– Ұзын тип (LONG);
lp	– Алыс нұсқағыш;
s	– Жол;
cz	– Нөль-символымен аяқталатын жол;
w	– Таңбасы жоқ бүтін (WORD);
x,y	– Қысқа тип(x немесе y координаты).

Мысалы, lpfnWndProc жазбасы lpfnWndProc-тің WndProc функциясына алыс нұсқағыш болатынын көрсетеді.

Келтірілген мысалдарда нұсқағыш түсінігі бірнеше рет ескерілді. Нұсқағыштарды пайдаланумен бағдарламалау технологиясы Windows жүйесін жазу кезінде кеңінен қолданылды. Сондықтанда бізге нұсқағыш түсінігін және онымен жұмыс істеу ережелерін ұғыну қажет. Бұл өте маңызды, өйткені біз қолданатын Windows жүйесінің API функцияларының көптеген формалды параметрлері нұсқағыштар болып табылады.

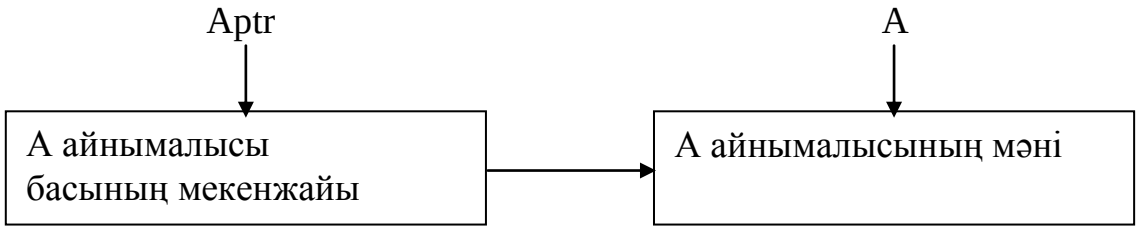
«Нұсқағыштар типтері негізінен C тілінің API-интерфейстерімен әрекеттескенде пайдалы болады, бірақ оларды басқарылатын шоғыр (куча) шегінен тыс болатын жадыға жүгіну үшін немесе бағдарламаның жоғары өнімді бөлімдерін жүзеге асырғанда пайдалануға болады» [Джозеф Албахари б. 171. C# 3.0 анықтамалығы]

3.2 Нұсқағыш түсінігі

Нұсқағыштар – бұл өз мәндері ретінде компьютер жадысының мекенжайын ұстайтын айнымалылар.

Осы тұрғыда айнымалының анықтамасын ескерту керет. Айнымалы – идентификатормен (айнымалы атауымен) белгіленген жады облысы, онда бағдарлама жұмысының барысында өзгерілетін деректер сақталынуы мүмкін. Айнымалы атауында осы айнымалының деректері орналасқан жадының мекенжайы жөнінде ақпарат болуға тиіс (әйтпесе біз өз деректерімізді компьютер жадысынан таппаймыз). Бірақ, қарапайым айнымалының мәні тұрған компьютер жадысының физикалық мекенжайы пайдаланушыға қол жетімсіз болса, ал айнымалы нұсқағыш компьютер жадысының физикалық мекенжайларымен, оның ішінде қарапайым айнымалы деректері тұрған жадының мекенжайларымен жұмыс істеу үшін арналған.

Яғни, егерде A айнымалы болса, A айнымалысының деректері бар жады мекенжайы орналасқан Aptr айнымалысы бар деп елестетуге болады. A айнымалысымен Aptr айнымалы-нұсқағышы арақатынасын келесі суретпен көрсетуге болады:



3.1-сурет – A айнымалысы және оның Aptr нұсқағышы

Нұсқағыштар кез келген басқа айнымалыларға ұқсас бағдарламадаға өзінің қолданылуының алдында хабардар етілуге тиіс. Нұсқағыштың хабардар етілу (сипатталу) форматы келесі түрде болады:

тип* айнымалы; мұнда
тип – мекенжайы айнымалыда сақталынатын жады облысындағы мәннің типі. Тип клас бола алмайды, бірақ құрылым, тізбелеу немесе нұсқағыш, сондай-ақ мәнді типтердің кез келгені және void бола алады. Void типі нұсқағыш белгісіз типті айнымалының мекенжайын сақтайтынын білдіреді. Нұсқағыштады сипаттаудың мысалдары:

```
int* aptr;
int  a;
```

біз бұл мысалда екі айнымалыны сипаттаймыз: a – бүтін типті қарапайым айнымалы; aptr –бүтін типті мәнге нұсқағыш.

float* xptr, yptr;

бұл жағдайда xptr және yptr айнымалылары нақты типті мәндерге нұсқағыштар болып табылады.

int*[] mas;

мысалда бүтін типті мәндерге нұсқағыштар массивін сипаттау нұсқасы қарастырылған.

int** iptr;

мысалда бүтін типті нұсқағышқа нұсқағыш сипатталған.

Айнымалыларды сипаттауда «*» символы оған сәйкес келетін айнымалы нұсқағыш болатынын көрсетеді (яғни мекенжайды сақтауға арналған). Нұсқағыштың мәні – бұл компьютер жадысының мекенжайы екенін тағы да ескертеміз.

3.3 Нұсқағыштың инициализациялау операциясы

Нұсқағыш бұл айнымалы, сондықтан да онымен кейбір операциялар жасауға болады.

Кез келген айнымалының басты операциясы инициализациялау операциясы (тағайындау) – айнымалыға қандайда бір мән тағайындалады. Нұсқағыш ерекше емес, бірақ мән ретінде компьютер жадысының мекенжайы алынуы тиіс. C# тілінде компьютер жадысының физикалық мекенжайын алу үшін арнайы операция бар – «&» адрестеу операциясы, ол өзі алдынан орнатылған айнымалының немесе басқа деректердің жадысының мекенжайын қайтарады. Мысалы,

Aptr = &a;

Осы мысалда Aptr нұсқағышына a айнымалысының мекенжайы тағайындалатын болады – a айнымалысының деректері сақталынатын компьютер жадысы облысының бастапқы мекенжайы. Мұнда Aptr a айнымалысы сияқты типтегі айнымалы нұсқағышы ретінде сипатталынуы тиіс.

Нұсқағыш мәніне басқа нұсқағыштың мәнін тағайындауға болады, бірақ тек типтері бірдей болу керек, мысалы, егерде Aptr және Bptr бүтін типті мәндерге нұсқағыштар болса және Bptr мәні анықталған болса (әйтпесе тағайындаудың мағынасы жоқ), онда келесі жазудың мағынасы бар болады:

Aptr = Bptr; .

C# тіліндегі айнымалыларды сипаттау олардың инициализациялауымен бірге жүруі тиіс, сондықтанда C# тілінде нұсқағыштар үшін компьютер жадысының жалған «нөлдік» мекенжайының константасы енгізілген - null. Кез келген типтегі нұсқағыштың мәні инициализациялау кезіңінде де, бағдарлама жұмысының барысында да компьютер жадысының осы «нөлдік» мекенжайына меншіктелінуі мүмкін, мысалы:

int* Aptr = null;

float Fptr;

Fptr = null;

Осы мысалда Aptr және Bptr нұсқағыштарға null деген компьютер жадысының жалған «нөлдік» мекенжайы тағайындалған.

Бағдарлама жұмысының барысында (немесе нұсқағыштардың инициализациясы кезінде) нұсқағыштарға айқын түрінде компьютер жадысының мекенжайын тағайындауға болады, бірақ бұл тағайындау тек осы мекенжай анық белгілі болса ғана жүргізіледі, өйткені олай болмаған жағдайда бағдарлама жұмысынан жаңылуы мүмкін, мысалы:

byte* Aptr = (byte *) 0x3F0ECD4;

char* Cptr = (char *) 0x3F0ECD4;

мұнда, 0x3F0ECD4 – жады мекенжайының он алтылық константасы, (byte *) немесе (char *) – константа нұсқағыштың сәйкес типіне келтірілетін типтерді келтіру операциялары.

Нұсқағыштар массивын инициализациялау және нұсқағыштардың көмегімен стекке жадыны бөлу мысалдарын нақты мысалдарда кейінірек қарастырамыз.

3.4 Нұсқағыштармен басқа операциялар

C# тілінде «*» символымен белгіленетін мәліметтер элементіне қол жеткізу /разыменование/ операциясы болады. Бұл операция адрестеу операциясына кері операция. Егерде адрестеу операциясы деректер бойынша олардың мекенжайын алса, ал мәліметтер элементіне қол жеткізу /разыменование/ операциясы мекенжайы бойынша деректерді алады. әрине, осы операцияға нұсқағыш қатысу керек. Мысалы,

a=*Aptr;

a айнымалысына мекенжайы Aptr нұсқағышында болатын деректер тағайындалады. Әрі Aptr a айнымалысы сияқты типтегі айнымалының нұсқағышы ретінде сипатталынуы тиіс.

C# тілінде құрылым типіндегі деректермен жұмыс кезінде «элементке нұсқағыш» деген арнайы операция немесе «->» символдарының комбинациясымен белгіленетін нұсқағыш арқылы құрылым элементіне қатынау операциясы бар. Мысалы, егерде PtrCtyd қандай да бір құрылымға нұсқағышты білдірсе, ал Name осы құрылымның өрісі болса, сонда PtrCtyd->Name жазбасы (*PtrCtyd).Name эквивалентті. Яғни *PtrCtyd мекенжайы бойынша біз бүкіл құрылымға жүгінеміз және онда Name өрісін таңдаймыз.

Нұсқағыштармен орындалатын негізгі операциялар 3.1-кестеде берілген.

3.1-кесте. Нұсқағыштармен орындалатын негізгі операциялар

Операция	Сипаттамасы
*	мәліметтер элементіне қол жеткізу /разыменование/ операциясы- нұсқағышта сақталған адресте орналасқан мәнді алу
->	Нұсқағыш арқылы құрылым элементіне қол жеткізу
[]	Нұсқағыш арқылы массив элементіне қол жеткізу
&	Айнымалы мекенжайын алу
++, --	Нұсқағыштың мәнін бір адрестелінетін элементке ұлғайту және азайту
+, -	Нұсқағыштарды бүтін шамалармен қосу және алу

==, !=, <>, <=, >=	Нұсқағышта сақталынған мекенжайларды салыстыру. Таңбасы жоқ бүтін шамалардың салыстыруы сияқты орындалады
Stackalloc	Стекте нұсқағыш сілтеп тұрған айнымалы үшін жады бөлу

3.5 Қауіпсіз емес код түсінігі

C# тілінің негізгі артықшылықтарының бірі – оның жадымен жұмыс істеу сызбасы: объектілер үшін жадыны автоматты түрде бөлу және қоқысты автоматты түрде тазалау. Әрі жадының қолданыста жоқ мекенжайына жүгінуге немесе массивтың шекараларынан асуға болмайды, бұл бағдарламаларды неғұрлым сенімді және қауіпсіз етеді және бір қатар кластардың қателерінің болуына жол бермейді.

Нұсқағыштар тікелей жады облысының мекенжайларымен жұмыс істеуге мүмкіндік береді, ал бұл C# тілінің қағидаттарына қарама-қайшы болады. Сондықтанда нұсқағыштарды қолданатын бағдарламалық код қауіпсіз емес деп аталатын болды.

Жалпы жағдайда CLR ортасы орындалуын бақыламайтын код қауіпсіз емес код деп аталынады.

CLR ортасымен дауды шешу үшін осындай код анық түрде unsafe негізгі сөзінің көмегімен белгіленуі керек. Осы сөз орындаудың қауіпсіз емес контекстін анықтайды.

Unsafe деген негізгі сөз класты, құрылымды немесе құрылым деректерінің өрісін сипаттау кезінде сипаттаушы ретінде қолданылуы мүмкін, мысалы:

public unsafe struct Ctyd {}.

Бұл жағдайда құрылымның барлық өрістері қауіпсіз емес ретінде белгіленеді. Егерде unsafe негізгі сөзі құрылымның тек бір өірісін сипаттау үшін сипаттаушы ретінде қолданылса, мысалы:

public struct Ctyd {… public unsafe int * Kol; … }

онда қауіпсіздік емес шарты осы құрылымның тек белгіленген өрісіне ғана таратылады.

unsafe негізгі сөзі күрделі оператор ретінде қолданылуы мүмкін, мысалы:

unsafe { блок }, мұнда белгіленген блокқа енген барлық операторлар қауіпсіз емес болады - яғни олардың жұмысы CLR ортасымен бақыланбайды және қауіпсіз емес айнымалыға бөлінген динамикалық жады C# тілінің қоқыс жинаушысымен «қорғалынбайды».

3.6 Нұсқағыштармен жұмыс бағдарламасының мысалы

Нұсқағыштармен жұмыс жасау ережесін жақсы түсіну және олармен операцияларды орындау үшін таза оқу мысалын қарастырамыз. Осы мысалда адрестеу және мәліметтер элементіне қол жеткізу /разыменование/ операцияларының жұмысын қарастырамыз.

Қауіпсіз емес фрагменті бар консолдық қосымша кодының жемісті компиляциясы үшін келесі жолмен Visual Studio ортасының баптауын өзгерту қажет: Project-> ConsoleAplication1 Properties-> Build командаларын таңдап, ашылған терезеде Allow Unsafe Code пунктінде «қанат белгі» («***птичка***») орнату

керек. Тереземен жұмысты Advanced батырмасы арқылы аяқтап, өзгертулерді OK батырмасын басумен растаймыз.

Қосымша кодында unsafe негізгі сөзін Main әдісінің сипаттауышы ретінде қолдануға болады, мысалы, `static unsafe void Main()`, сонда осы әдістің барлық операторлары қауіпсіз емес ретінде қарастырылатын болады.

Бағдарламаның бастапқы коды (кодтың кейбір фрагменттері Павловскаяның кітабынан алынған):

```
using System;

namespace ConsoleApplication1
{
 class Program
 {
  static unsafe void Main()
  {
   int a, c;
   uint adr;
   a = 10;
   int* aPtr = null;
   aPtr = &a;
   Console.WriteLine("*aPtr = " + *aPtr);
   Console.WriteLine("a = " + a);
   Console.Write("aPtr = ");
   adr = (uint)aPtr;
   byte* b = (byte*)&adr;
   b = b + 3;
   Console.Write("0x");
   for (int i = 0; i < 4; i++)
       { Console.Write("{0:X}", *b); b--; }
   Console.WriteLine();
   Console.Write("&a   = ");
   adr = (uint)&a;
   b = (byte*)&adr;
   b = b + 3;
   Console.Write("0x");
   for (int i = 0; i < 4; i++)
       { Console.Write("{0:X}", *b); b--; }
   Console.WriteLine();
   Console.WriteLine("adr = " + adr);
   c = *& a;
   Console.WriteLine("*& a = " + c);
   //c = &* a;
   //Console.WriteLine("&* a = " + c);
   Console.Write("&* aPtr = ");
   adr = (uint)&* aPtr;
   b = (byte*)&adr;
   b = b + 3;
   Console.Write("0x");
   for (int i = 0; i < 4; i++)
       { Console.Write("{0:X}", *b); b--; }
   Console.WriteLine();
   Console.Write("*& aPtr = ");
   adr = (uint)*& aPtr;
   b = (byte*)&adr;
```

```
    b = b + 3;
    Console.Write("0x");
    for (int i = 0; i < 4; i++)
        { Console.Write("{0:X}", *b); b--; }
    Console.WriteLine();
    Console.ReadKey();
   }
  }
}
```

Работа программы:

*aPtr = 10
a = 10
aPtr = 0x3F0ECD4
&a = 0x3F0ECD4
adr = 66120916
*& a = 10
&* aPtr = 0x3F0ECD4
*& aPtr = 0x3F0ECD4

Мәліметтер элементіне қол жеткізу /разыменование/ және адрестеу сияқты екі қатар операциялары нұсқағыш үшін де, солай тұтас типті айнымалы үшін де өзара бір-бірін жояды (*& a = 10 қараңыз).

Мәліметтер элементіне қол жеткізу /разыменование/ және адрестеу сияқты екі қатар операциялары тек нұсқағыш үшін ғана өзара бір-бірін жояды. Тұтас типті айнымалы үшін осы комбинацияны қолдану әрекеті Error 1 The * or -> operator must be applied to a pointer деген хабарламамен бағдарлама компиляциясының қатесіне әкеледі.

Нұсқағыштармен жұмыс кезінде егерде нұсқағыштар бағдарламаның басында инициализацияланатын болса, бұл жақсы стиль деп саналады, яғни оларға немесе кейбір мәндер тағайындалса, немесе нұсқағыштарға null-ға тең нөлдік мәндер берілсе.

3.7 void* типі үшін нұсқағышты пайдалану

Нұсқағышты сипаттау кезінде void* типін пайдалануға болады, бұл кейінге қалдырған типті нұсқағышты сипаттауды білдіреді.

Осындай void* типіндегі нұсқағыш үшін арифметикалық операцияларды және мәліметтер элементіне қол жеткізу /разыменование/ операцияларын қолдануға болмайды.

Бірақ,ол void* типті нұсқағыштың кез келген типті нұсқағышқа анық емес түрлендірілуі үшін қолдланылуы мүмкін. Сол уақытта void* типіндегі нұсқағышқа кез келген типтегі айнымалының мекенжайын тағайындауға болады. Сонымен, void* типіндегі нұсқағышты кез келген типті айнымалылардың мекенжайларының аралық сақтауышысы ретінде пайдалануға болады. Таза оқу мысалын қарастырамыз, онда void* типіндегі нұсқағышқа кезек-кезегімен бүтін және нақты типті айнымалыларының мекенжайларын тағайындайтын боламыз, бұдан соң осы мекенжайларды тиісінше бүтін және нақты типтер үшін нұсқағыштарға тапсырамыз.

```
using System;

namespace ConsoleApplication1
{
 class Program
 {
  static unsafe void Main()
  {
   int a = 10;
   double b = 15.67;
   int* aPtr = null;
   double* bPtr = null;
   void* cPtr=null;
   for (int i = 1; i <= 5; i++)
   {
    if (i % 2 == 0)
    {
     cPtr = &a; aPtr = (int*)cPtr;
     Console.WriteLine("i = {0} По адресу указателя находится число
{1} ", i, *aPtr);
    }
    else
    {
     cPtr = &b; bPtr = (double*)cPtr;
     Console.WriteLine("i = {0} По адресу указателя находится число
{1} ", i, *bPtr);
    }
   }
   Console.ReadKey();
  }
 }
}
```

Работа программы:

i = 1 По адресу указателя находится число 15,67
i = 2 По адресу указателя находится число 10
i = 3 По адресу указателя находится число 15,67
i = 4 По адресу указателя находится число 10
i = 5 По адресу указателя находится число 15,67