Programozás II.

2. gyakorlat

Szerémi Éva Krisztina

Témák

- Mutató
- Vektor
- Dinamikus változó
- Dinamikus tömb
- Parancssori argumentum

Mutató - példa

```
#include<iostream>
using namespace std;
int main()
{
      int f = 10 ;
      int *mut;  // intre mmutató mutató típusú változó létrehozása
      int &fi=f; //hivatkozási típusú változó létrehozása,
      mut=&f;    //mut pointer is f-re mutat
      cout<<"f="<<f<<" fi="<<fi<< " *mut= "<<*mut<<endl;
      fi=20;
      cout<<"f="<<f<<" fi="<<fi<< " *mut= "<<*mut<<endl;
      cout << &f << " "<<&fi << " " << mut<<" "<<&mut << endl;
                  // az első 3 ugyanoda mutat a 4. nem (a mutató címe)
      system("pause");
      return 0;
}
```

Vektor - példa

Kérjük be egy minimum 3 fős osztály tanulóinak jegyeit egy vektorba úgy, hogy nem tudjuk, hányan járnak az osztályba (addig kérjük a jegyeket, amíg van adat).

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int main()
{
	vector <int> jegyek(3); // 3 elemű vektor
	char bet;
	int szam;
	int i;
	for (i = 0;i < 3; i++)
	{
		cout << "Adja meg az " << i + 1 << ". tanulo jegyet(1-5) ";
	cin >> jegyek[i];
	}
```

Vektor - példa

```
cout << "akar meg jegyet felvenni az osztalyba? ( i=> igen )" << endl;
cin >> bet;
if (bet == 'i’)
{
      do{
            cout << "Adja meg az " << i + 1 << ". tanulo jegyet(1-5) ";
            cin >> szam;jegyek.push_back(szam); //hozzáfűzünk egyjegyet
            cout << "akar meg jegyet felvenni: ( i=> igen )";
            cin >> bet;
            i++;
      } while (bet == 'i’);
}
cout << "Jegyek szama " << jegyek.size()<<endl;
cout << "Az osztaly jegyei : " << endl;
for (i = 0; i < jegyek.size(); i++)
      {
            cout << jegyek.at(i)<<" ";   // jegyek[i]
      }
```

Vektor - példa

```
	jegyek.pop_back();// törli az utolsó elemet
	cout << "\nAz osztaly jegyei az utolso elem torlese utan : " << endl;
	for (i = 0; i < jegyek.size(); i++)
	{
		cout << jegyek.at(i) << " ";   // jegyek[i]
	}
	system("pause");
	return 0;
}
```

Dinamikus változók - tömbök

Létrehozás: new

Felszabadítás: delete

Pl.:

```
int *szam;                  // mutató létrehozása
szam = new int;             //a memória címe belekerül szam-ba
*szam = 20;                 //értékadás a din. változónak
cout << "szam=" << *szam << endl;
delete szam;                //a dinamikus változó felszabadítása
```

```cpp
#include <iostream>
using namespace std;
int main()
{
	cout << "*** Hagyomanyos (statikus) memoria kezeles ***" << endl;
			//statikus (hagyományos) tömbök használata
	int jegy[5] = { 1,2,3 };			// 1, 2, 3,0,0
	cout << "int jegy[5]={1,2,3}; tomb merete=" << sizeof(jegy) << endl;
		//a sizeof megadja a tömb méretét => elemeinek számát*4 bájt(int)
	cout << "jegy[0]=" << jegy[0] << endl;   //hivatkozás a 0 indexű elemre
	cout << "jegy[2]=" << *(jegy + 2) << endl;     //a jegy kezdőcímétől().
				elemtől lépjünk kettőt, ez a 2-es indexű elem
	cout << "***  DINAMIKUS MEMORIA KEZELES  ***" << endl;
	int *szam;		//mutató a dinamikus változóhoz, ez a verembe kerül
	szam = new int;		//dinamikus változó definíció
	if (szam == NULL) //ha NULL érték kerül a változóba nem egy memória cím
	{
		cerr << "hiba:keves a memoria";
		system("pause");
		return 1;
	}			//a memóriacíme belekerül iptr-be
```

```
*szam = 20;                  //értékadás a dinamikusváltozónak
cout << "szam=" << *szam << endl;
delete szam;                 //a dinamikus változó felszabadítása,
//dinamikus tömb használata, a halomban kerül tárolásra
int db;
cout << "***  dinamikus tomb hasznalata  ***" << endl;
cout << "a dinamikus tomb elemszama: n=";
cin >> db;
int *tmb = new int[db]; //db elemű dinamikus int típusú tömb definíció,
if (tmb == NULL)
              //a tömb neve 'tmb' egy mutató típusú váltózó, azt a
              memóriacímet jelenti a dinamikus memóriában, ahol a tömb
              kezdődik a memóriában
{             //az elemszám változó is lehet, tehát nem kell ismerni a
              programírásakor.
      cout << "hiba: keves a memoria";
      system("pause");
      return 1;
}
```

```
    cout << "sizeof(tmb): a mutato merete = " << sizeof(tmb) << endl;
                //ez nem a tömb, hanem a mutató méretét adja meg
    *tmb = 15;          //értékadás a din. tömb0 indexű elemének tmb[0]=15
    tmb[3] = 10;        //értékadás a din. tömb 3 indexű elemének
    cout << "tmb[0]=" << tmb[0] << endl;
    *(tmb + 1) = 20;                  //értékadás a din. tömb 1 indexű elemének
    cout << "tmb[1]=" << *(tmb + 1) << endl;
    delete []tmb;
          //a dinamikus tömb felszabadítása, fontos [] használata, példa -
          döntsük el, hogy van-e értelme az alábbi programrészletnek hibás-
          e?, a tomb most statikus tomb
    int tomb[5] = { 1,2,3,4,5 };
    int i =  3;
    cout << tomb[i] << endl;              // *(tomb+i)
    cout << i[tomb] << endl;              //*( i+tomb)
    cout << &tomb[i] << endl;
    cout << &i[tomb] << endl;
    cout << "tmb[3]=" << *(tmb + 3) << endl;
                //hiba, mert már felszabadítottuk a memóriát
    system("pause");
    return 0;
}
```

Parancssori argumentum használata

Készítsünk programot, amely összeszoroz két számot, de ne billentyűzetről kérjük be a számokat, hanem parancssori argumentumból kapja!

Argumentumok beállítása a fejlesztő környezetben:
Project Menü -> (consolapplic..) Properties -> Debugging -> Command arguments (ide kell beírni szóközökkel elválasztva az argumentumokat)

```
#include <iostream>
#include <fstream>
using namespace std;
int main (int argdb, char *argtomb[])
{
	/* HAGYOMÁNYOS PROGRAM: double a,b; cout<<"adjon meg ket szamot";
		cin>>a>>b; cout<<a*b;*/
	if(argdb!=3)
	{
		cerr<<"Hiba!!!\nInditas helyesen: "<<argtomb[0]<<"szam1 szam2\n";
		system("pause");
		return 1;
	}
	double a,b;
	a=atof(argtomb[1]);		//sztringből atof() float-ot
	b=atof(argtomb[2]);
	cout<<a<<" * "<<b<<" = "<<a*b<<endl;
	system("pause");
	return 0;
}
```