PROGRAMI I ORIENTUAR NE OBJEKTE

Variabla të tipit tregues - Pointer

POINTERËT

- Kompjuteri gjatë operimit me të dhëna i shfrytëzon adresat e lokacioneve në të cilat ato janë vendosur, përkatësisht adresat e variablave përkatëse.
- Por, që edhe programuesi të ketë qasje në këto adresa, në gjuhën C++ shfrytëzohen variabla të tipit tregues, ose, siç thuhet ndryshe - variabla pointer (ang. pointer), ose shkurt vetëm pointer.

POINTERËT

- Këto variabla quhen kështu, sepse përmes tyre merren adresat e variablave, përkatësisht vlerat që ruhen brenda tyre tregojnë te variablat.
- Pointerët, në raste të caktuara kanë një përparësi, sepse përmes adresave, pa kufizime, mund të shfrytëzohen vlerat e variablave në pjesë të ndryshme të programit.
- Për t'i dalluar nga variablat e zakonshme, para pointerëve shënohet simboli * , psh.
  - int *a;

ADRESAT E VARIABLAVE

- Duke e shfrytëzuar operatorin &, përmes shprehjes së formës:
  - p=&v;
- Adresa e variablës së zakonshme v vendoset te variabla p e tipit pointer.

MARRJA DHE SHTYPJA E ADRESËS SË VARIABLËS X, DUKE E SHFRYTËZUAR POINTERIN A.

PRISHTINË

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5   int *a,x;
6   x=357;
7   cout << "\nVlera e variablës x: "
8        << x
9        << "\n";
10  a=&x;
11  cout << "\nAdresa te pointeri a: "
12       << a
13       << "\n\n";
14  cin.get();
15  return 0;
16 }
```

VLERA NË ADRESËN E VARIABLËS

- Nëse dihet adresa e një variable, përmes procesit të deadresimit ose të dereferimit (ang. dereferencing), mund të merret vlera e variablës. Për këtë qëllim shfrytëzohet operatori *, kështu:
  - v=*p;
- pas së cilës, te variabla v do të vendoset vlera e variablës adresa e së cilës ruhet te pointeri p.

SHTYPJA E VLERËS SË VARIABLËS NDËRMJETËSUESE H, DUKE E SHFRYTËZUAR ADRESËN E VARIABLËS G, E CILA RUHET TE POINTERI A.

```
 1  #include <iostream>
 2  using namespace std;
 3  int main()
 4  {
 5      int *a,g,h;
 6   g=357;
 7   cout << "\nVlera e variablës g: "
 8        << g
 9        << "\n";
10   a=&g;
11   cout << "\nAdresa te variabla a: "
12        << a
13        << "\n";
14   h=*a;
15   cout << "\nVlera e variablës h: "
16        << h
17        << "\n\n";
18   cin.get();
19   return 0;
20  }
```

PASS-BY-VALUE WITH POINTERS

PRISHTINË

```
1  #include <iostream>
2  using namespace std;
3
4  int cubeByValue( int ); // prototype
5
6  int main()
7  {
8      int number=5;
9      cout << "The original value of number is "<< number;
10     number =cubeByValue( number ); // pass number by value to cubeByValue
11     cout << "\nThe new value of number is " << number << endl;
12     cin.get();
13     cin.get();
14 } // end main
15
16 // calculate and return cube of integer argument
17 int cubeByValue( int n)
18 {
19     return n*n*n; // cube local variable nand return result
20 } // end function cubeByValue
```

PASS-BY-REFERENCE WITH POINTERS

PRISHTINË

```
1  #include <iostream>
2  using namespace std;
3
4  void cubeByReference( int * ); // prototype
5
6  int main()
7  {
8      int number=5;
9      cout << "The original value of number is "<< number;
10     cubeByReference( &number ); // pass number by reference to cubeByReference
11     cout << "\nThe new value of number is " << number << endl;
12     cin.get();
13     return 0;
14 } // end main
15
16 // calculate cube of *nPtr; modifies variable number in main
17 void cubeByReference( int *nPtr)
18 {
19     *nPtr = *nPtr * *nPtr * *nPtr; // cube *nPtr
20 } // end function cubeByReference
```

SIZEOF OPERATOR

PRISHTINË

```
1  // Size of operator when used on an array name
2  // returns the number of bytes in the array.
3  #include <iostream>
4  using namespace std;
5
6  size_t getSize(double * ); // prototype
7
8  int main()
9  {
10     double array[ 20 ]; // 20 doubles; occupies 160 bytes on our system
11     cout << "The number of bytes in the array is " << sizeof(array );
12     cout << "\nThe number of bytes returned by getSize is "
13     <<getSize( array) << endl;
14     cin.get(); return 0;
15 } // end main
16
17 // returnsizeofptr
18 size_t getSize( double *ptr)
19 {
20     return sizeof(ptr );
21 } // end function getSize
```

POINTERËT NË ANËTARËT E VEKTORËVE

- brenda unazës for është vendosur shprehja:
  - a=&Z[i];
- përmes së cilës adresa e anëtarit të i-të të vektorit vendoset te pointeri a.

SHFRYTËZIMI I POINTERIT A GJATË SHTYPJES SË ANËTARËVE TË VEKTORIT Z(N).


```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main()
5  {
6      const int n=5;
7      int *a,i,Z[n]={27,12,-23,9,35};
8      char h[]="-----------------------------";
9   cout << "\n Indeksi   Anëtari   Adresa\n"
10       << h
11       << endl;
12
13  for (i=0;i<n;i++)
14  {
15      a=&Z[i];
16      cout << setw(6) << i << setw(10) << Z[i] << setw(12) << a << endl;
17  }
18
19  cout << h
20       << "\n";
21  cin.get();
22  return 0;
23 }
```

```cpp
// Using subscripting and pointer notations with arrays.
#include <iostream>
using namespace std;
int main()
{
    int b[]={10, 20, 30, 40 }; // create 4-element array b
    int *bPtr =b;// set bPtr to point to array b

    // output array b using array subscript notation
    cout << "Array b printed with:\n\nArray subscript notation\n";

    for ( int i=0;i<4;++i )
        cout << "b[" << i<<"] ="<<b[i] << '\n';

     // output array b using the array name and pointer/offset notation
    cout << "\nPointer/offset notation where "
        << "the pointer is the array name\n";
    for ( int offset1 = 0; offset1 < 4; ++offset1 )
        cout << "*(b+"<< offset1 << ") ="<< *( b + offset1) << '\n';

    // output array b using bPtrand array subscript notation
    cout << "\nPointer subscript notation\n";
    for ( int j=0;j<4;++j )
        cout << "bPtr[" << j<<"] ="<<bPtr[j] << '\n';
    cout << "\nPointer/offset notation\n";

    // output array b using bPtrand pointer/offset notation

    for ( int offset2 = 0; offset2 < 4; ++offset2 )
        cout << "*(bPtr +"<< offset2<<") ="
        << *( bPtr + offset2 ) << '\n';
    cin.get();
} // end main
```

PRISHTINË