**Pointeri si referinte**

In C++ exista doua modalitati de lucra cu adrese de memomorie: pointeri si referinte.

**Pointeri**

Pointerii sunt variabile care contin adresa unei alte zone de memorie. Ei sunt utilizati pentru a memora date care sunt cunoscute prin adresa zonei de momorie unde sunt alocate.

Sintaxa utilizata pentru declararea lor este:

**tip \*variabila\_pointer;**

Exemplu:

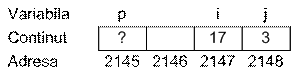
      // declaratie variabile

      int i = 17, j = 3;

      // declaratie pointer

      int \*p;

Continutul memoriei in urma acestor declaratii va fi:



Se observa ca pointerul la acest moment nu este initializat. Referirea prin intermediul pointerului neinitializat va genera o eroare la rularea programului.

In lucrul cu pointeri se folosesc doi operatori unari:

* **&**: extragerea adresei unei variabile
* **\***: referirea continutului zonei de memorie indicate de pointer (indirectare)

Exemplu:

|  |  |
| --- | --- |
| // p ia adresa lui i  p = &i; | http://ase.softmentor.ro/LimbajeEvoluate/02_ReferinteSiPointeri_files/image004.gif |
| // modificarea  // continutul zonei  // de memorie  // pointate de p  (\*p) = 6; | http://ase.softmentor.ro/LimbajeEvoluate/02_ReferinteSiPointeri_files/image006.gif |

Un pointer poate fi refolosit, in sensul ca poate contine adrese diferite la diferite momente de timp:

|  |  |
| --- | --- |
| // modificare adresa  p = &j; |  |

Operatiile permise asupra pointerilor sunt urmatoarele:

* extragerea obiectului referit de catre pointer folosind operatorul **\***sau operatorul [] (prezentat in sectiunea **Masive**)
* extragerea adresei unui pointer folosind operatorul **&** (se va obtine un pointer la pointer)
* atribuirea intre doi pointeri care refera acelasi tip de data
* incrementarea/decrementarea (va muta pointerul inainte/inapoi cu un numar de bytes egal cu dimensiunea tipului referit)
* adunarea/scaderea cu o valoare intreaga (va muta pointerul inainte/inapoi cu un numar de bytes egal cu dimensiunea tipului referit inmultita cu valoarea intreaga)
* diferenta a doi pointeri de acelasi tip (se obtine numarul de elemente de tipul respectiv ce incap intre cei doi pointeri)
* compararea a doi pointeri
* conversia pointerilor (se realizeaza ca si pentru celelalte tipui folosind operatorul de cast)

**Referinte**

Referintele, ca si pointerii, sunt variabile care contin adresa unei zone de memorie. Semantic, ele reprezinta aliasuri ale unor variabile existente.

Referintele sunt legate de variabile la declaratie si nu pot fi modificate pentru a referi alte zone de memorie. Sintaxa folosita pentru declararea unei referinte este:

**Tip & referinta = valoare;**

Exemplu:

|  |  |
| --- | --- |
| // declaratii variabile  int i = 6, j = 3;    // declaratie referinta  int& r = j; | http://ase.softmentor.ro/LimbajeEvoluate/02_ReferinteSiPointeri_files/image008.gif  r |

Sintaxa utilizata pentru manipularea pointerului este aceeasi adică cea a variabilei de care este legata (indirectarea este realizata automat de catre compilator). Toate modificarile aplicate referintei se vor reflecta asupra variabilei referite.

Exemple:

|  |  |
| --- | --- |
| // modificarea variabilei  // prin referinta  r = i; |  |
| // atribuirea are ca efect  // copierea continutului  // din i in j si nu  // modificarea adresei referintei | http://ase.softmentor.ro/LimbajeEvoluate/02_ReferinteSiPointeri_files/image010.gif |

Spre deosebire de pointeri, referintele nu au operatii speciale. Toti operatorii aplicati asupra referintelor sunt de fapt aplicati asupra variabilei referite. Chiar si extragerea adresei unei referinte va returna adresa variabilei referite. Pentru exemplul prezentat, expresia **&r** va returna valoarea 2148 (operatorul de extragere de adresa se aplica de fapt asupra variabilei **j**).

Proprietatile cele mai importante ale referintelor sunt:

* referintele trebuie sa fie initializate la declaratie (spre deosebire de pointeri care pot fi initializati in orice moment);
* dupa initializare, referinta nu poate fi modificata pentru a referi o alta zona de memorie (pointerii pot fi modificati pentru a referi alta zona)
* intr-un program C++ valid nu exista referinte nule

Referintele utilizate in principal pentru transmiterea parametrilor in functii.

**Trimiterea parametrilor in functii**

Trimiterea parametrilor in functii se poate face prin doua mecanisme:

* **prin valoare**: valorile parametrilor sunt copiate pe stiva; modificarile efectuate de functie asupra parametrilor nu se vor reflecta in apelant
* **prin adresa**: se copiaza pe stiva adresele de memorie unde se afla datele corespunzatoare parametrilor; modificarile efectuate de functie vor fi vizibile si in apelant

Transferul prin adresa se poate realiza prin intermediul pointerilor sau referintelor. Recomandarea generala este sa sa foloseasca referintele datorita sintaxei mai simple si a faptului ca permit evitarea unor probleme specifice pointerilor (pointeri nuli, pointeri catre zone dezalocate, …).

Exemple de transmitere parametri:

|  |  |
| --- | --- |
| **Functie** | **Apel** |
| Prin valoare:  void Inc(int i)  {        i++;  } | int i = 10;  Inc(i);  cout << i;    Rezultat: 10 |
| Prin referinta:  void IncReferinta(int &i)  {        i++;  } | int i = 10;  IncReferinta(i);  cout << i;    Rezultat: 11 |
| Prin pointeri:  void IncPointer(int \*pi)  {        (\*pi)++;  } | int i = 10;  IncPointer(&i);  cout << i;    Rezultat: 11 |

**Pointeri si referinte constante**

Modificatorul **const** poate fi folosit pentru a declara pointeri constanti sau pointeri la zone constante.

Pointerii constanti sunt pointeri care nu-si pot modifica adresa referita. Sintaxa folosita este:

**tip \* const pointer\_ct = adresa;**

In cazul pointerilor constanti, initializarea la declarare este obligatorie.

Exemplu:

       // declarare si initializare

       // pointer constant

       int \* const pConstant = &i;

       // modificarea continutului este permisa

       (\*pConstant) = 5;

       // modificarea adresei nu este permisa

       pConstant = &j; // => eroare de compilare

Pointerii la zone de momorie constante sunt pointeri prin intermediul carora nu se poate modifica continutul zonei referite.

Sintaxa de declarare este:

**tip const \* pointer\_zona\_ct;**

Exemplu:

       // declarare pointer la zona constanta

       int const \* pZonaCt;

       // modificarea adresei referite

       // este permisa

       pZonaCt = &i;

       pZonaCt = &j;

       // modificarea continutului nu este permisa

       (\*pZonaCt) = 7; // => eroare de compilare

Cele doua forme pot fi folosite simultan pentru a declara pointeri constanti la zone de memorie constanta:

**tip const \* const pointer\_ct\_zona\_ct;**

In acest caz nu poate fi modificata nici adresa referita, nici constinutul acesteia.

Referintele sunt implicit echivalente cu un pointer constant (adresa referita nu poate fi modificata). In cazul lor, modificatorul **const** poate fi utilizat pentru a crea referinte prin intermediul carora nu se pot efectua modificari asupra continutului. Sintaxa utilizata este:

**tip const& referinta\_ct;**

Comportamentul este echivalent cu al pointerilor constanti la zone de memorie constanta (referinta va putea fi utilizata numai pentru citirea valorii referite).