Pointerii

Au colaborat :
Sava Vladislav
Platon Alexandru

Pointerul este o variabila care contine o adresa de memorie . Aceasta variabila contine adresa unei variabile . Avantajele utilizarii pointerilor sunt :

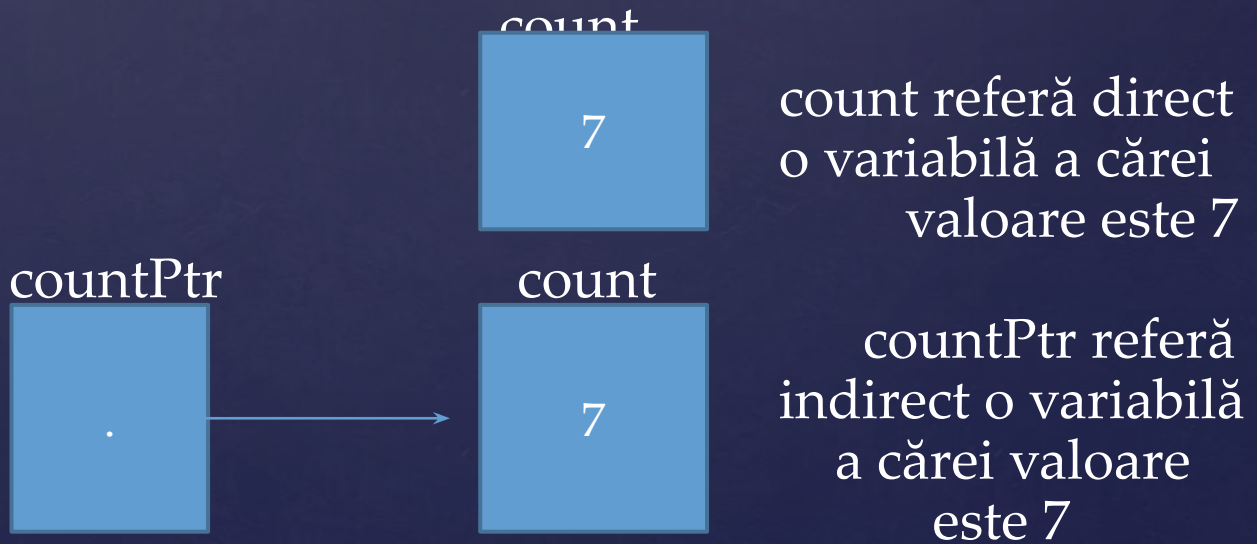
- Ofera posibilitatea de a modifica argumentele de apelare a functiilor ;
- permite o alocare dinamica a memoriei;
- Pot imbunatati eficienta anumitor rutine.

Pointerii reprezinta una din cele mai puternice caracteristici ale limbajului C , dar si periculoase . Daca pointerii nu sunt initializati corect sau daca contin valori incorecte pot determina blocarea calculatorului , sau sa conduca la erori greu de depistat .

Variabilele de tip Pointer :

Variabilele de tip pointer stochează adrese de memorie. Pot, de exemplu, să păstreze adrese de memorie ale altor variabile care, la rândul lor, conţin alte valori. În acest sens, un nume de variabilă referă direct o valoare, iar un pointer referă indirect o valoare. Referirea unei valori printr-un pointer se numeşte indirectare.

Pointerii ca orice alta variabila trebuie declarati inainte de a fi folositi .



Exemplu:

1) int *countPtr, count;

Prin aceste declaraţii, variabila countPtr este de tip int*, adică este pointer către o valoare întreagă. Variabila count este de tip întreg şi nu pointer la întreg. Fiecare variabilă declarată ca pointer este precedată de un asterisc *.

2) double *x, *y;

Atât x cât şi y sunt pointeri către valori de tip double. Aceste variabile pot păstra adrese de memorie ale unor valori de tip double. Pot fi declaraţi pointeri ca să pointeze către variabile de orice tip de dată. Este indicat ca pointerii să fie iniţializaţi fie odată cu declaraţia acestora, fie printr-o instrucţiune de asignare. Un pointer poate fi iniţializat cu 0, NULL sau cu o adresă de memorie. Un pointer cu valoarea 0 sau NULL nu pointează către nicio zonă de memorie. Constanta NULL este declarată în fişierul header şi în alte câteva fişiere din biblioteca standard. Iniţializarea prin valoarea NULL este echivalentă cu iniţializarea prin valoarea 0, dar în C++ se preferă cea de-a doua variantă. Întregul 0 este convertit automat către o adresă de tipul pointerului.

Operatori pentru pointeri

Operatorul adresă & este unar şi returnează adresa operandului său. Exemplu int y = 5; int *yPtr; yPtr = &y; Prin ultima instrucţiune, adresa de memorie a variabilei y este încărcată în variabila pointer yPtr. În urma acestei asignări, vom spune că yPtr pointează către y.

Exemplu:

```
 #include <iostream>
using std::cout;
 using std::endl;
 int main()
 {
 int a;
int *aP;
a = 7;
 aP = &a;
 cout << "Adresa lui a este " << &a
          << "\nValoarea lui aP este " << aP;
cout << "\n\nAdresa lui a este " << a
        << "\nValoarea lui *aP este " << *aP;
 cout << "\n\nOperatorii * si & sunt inversi unul altuia. “
        << "\n&*aP = " << &*aP
         << "\n*&aP = " << *&aP << endl;
 cout << "\n\nAdresa lui aP este " << &aP << endl;
 return 0;
 }
```

Acest program afişează pe ecran următorul rezultat:

Adresa lui a este 0x22ff74
Valoarea lui aP este 0x22ff74
Adresa lui a este 7
Valoarea lui *aP este 7
Operatorii * si & sunt inversi unul altuia.
&*aP = 0x22ff74 *&aP = 0x22ff74
Adresa lui aP este 0x22ff70

Pointeri şi tablouri

Tablourile şi pointerii sunt, în limbajul C++, în strânsă legătură. Un nume de tablou poate fi interpretat ca un pointer constant, iar pointerii pot fi indexaţi ca şi tablourile.

Pentru tabloul v[5] am declarat variabila pointer vPtr pe care am iniţializat-o cu v, adresa primului element al tabloului. Elementul v[3] poate fi referit şi prin expresiile pointer

*(vPtr + 3)

*(v + 3)

Valoarea 3 din aceste expresii se numeşte offset la pointer, iar o astfel de expresie care accesează un element al unui tablou se numeşte notaţie offset sau notaţie pointer. Fără paranteze, expresia

*vPtr + 3

ar fi adunat valoarea 3 la expresia *vPtr, adică la v[0]. Pentru pointeri se pot folosi indici la fel ca şi pentru tablouri.