Ponteiros e Passagem por referência

Ponteiros

Um **ponteiro** é uma variável que armazena o endereço de memória de outra variável.
Declaração: `tipo *nome_ponteiro;`
Ex: `int *p;`

Operador `&`: forneçe o endereço de memória de uma variável.
Ex.:

```
int a = 1;
int *p;
p = &a;
```

Operador `*`: acessa variável cujo endereço é fornecido pelo ponteiro.
Ex.:

```
int b = 2 + *p; // b recebe valor 3 (2 + 1)
*p = 5; // isto modifica a variável 'a'
```

Esquema ilustrativo da memória:

```
int a = 1;
int *p;
p = &a;

 Endereço  Valor  Variável
          -------
      100 |     |
          -------
      200 |  1  | a
          -------
      300 |     |
          -------
      400 |     |
          -------
      500 |     |
          -------
      600 | 200 | p
          -------
      700 |     |
          -------
            ...
```

```
int b = 2 + *p; // b recebe valor 3 (ou seja, 2 + 1)
*p = 5; // isto modifica a variável 'a'

Endereços  Valor  Variável
           -------
      100 |     |
           -------
      200 |  5  | a
           -------
      300 |     |
           -------
      400 |  3  | b
           -------
      500 |     |
           -------
      600 | 200 | p
           -------
      700 |     |
           -------
             ...
```

Teste o código abaixo:

```
#include <iostream>
using namespace std;

int main() {
    int a = 1;
    int *p = &a;
    cout << "Endereço de memória de a: " << p << endl;
    cout << "Valor armazenado no local apontado: " << *p << endl;
    *p = 5;
    cout << "Valor armazenado no local apontado: " << *p << endl;
    cout << "Valor de a: " << a;
    return 0;
}
```

Aplicação de ponteiros: passagem por referência

Vimos que uma função não modifica um inteiro ou ponto flutuante passado como parâmetro:

```
void f(int x)
{
   x = 2;
}

int main()
{
   int y = 1;
   f(y);
   cout << y;  // Imprime o valor 1.
   return 0;
}
```

Para uma função modificar o valor passado como parâmetro, devemos passar o endereço de memória da variável que deve ser modificada (ponteiro). Chamamos este procedimento de passagem de parâmetro por referência.
Teste o código abaixo:

```
#include <stdio.h>
using namespace std;

void f(int *x)
{
   *x = 2;
}

int main()
{
   int y = 1;
   f(&y);
   cout << y;  // Imprime o valor 2.
   return 0;
}
```

Ponteiros para estruturas

Podemos ter ponteiros para estruturas. Portanto, podemos modificar campos de estruturas passadas como parâmetro.
Teste o código abaixo:

```
#include <iostream>
#include <iomanip>
using std::cout;

struct tupla {
   int x;
   float y;
};

void f(tupla *p)
{
   (*p).y = 3;
}

int main()
{
   tupla t = {1, 2.0};
   f(&t);
   cout << std::fixed;
   cout << t.x << std::setprecision(1) << t.y;  // Imprime "1 3.0"
   return 0;
}
```

Para melhorar a legibilidade, a linguagem C++ permite escrever `(*t).x` como `t->x`.

```
#include <iostream>
#include <iomanip>
using std::cout;

struct tupla {
   int x;
   float y;
};

void f(tupla *p)
{
   p->y = 3;  // ALTERE APENAS AQUI.
}

int main()
{
   tupla t = {1, 2.0};
   f(&t);
   cout << std::fixed;
   cout << t.x << std::setprecision(1) << t.y;  // Imprime "1 3.0"
   return 0;
}
```

Identificadores de vetores são ponteiros

Quando um vetor é declarado, o nome do vetor é tratado como um ponteiro. Internamente, a compilador reserva um espaço contíguo de memória capaz de armazenar todos os elementos do vetor, e faz o nome da variável ser um ponteiro para o início deste espaço de memória. Isto implica que todo vetor passado como parâmetro para uma função na verdade é passado por referência: se a função alterar o vetor, esta mudança ocorrerá no vetor passado como parâmetro.
Teste o código abaixo:

```
#include <iostream>

void f(int v[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      v[i] += 1;
}

int main()
{
   int v[3] = {1,2,3};

   f(v,3);

   int i;
   for (i = 0; i < 3; i++)
      std::cout << v[i] << " "; // Imprime "2 3 4"
   return 0;
}
```

Última atualização: domingo, 29 nov 2020, 13:06

◄ [struct, array] Busca em Vetor de Estruturas

Seguir para...

Alocação dinâmica de memória em C++ ►

Av. José de Freitas Queiroz, 5003
Cedro – Quixadá – Ceará CEP: 63902-580
Secretaria do Campus: (88) 3411-9422