Ponteiros e Alocação Dinâmica

Roland Teodorowitsch

Programação Orientada a Objetos - ECo - Curso de Engenharia de Computação - PUCRS

11 de outubro de 2023

Ponteiros em C++

Revisão

- O operador * antecedendo o nome de uma variável em uma declaração, define um ponteiro o respectivo tipo

```
double *mediaPtr;
```

- Operador de referência (&): do lado esquerdo de uma variável, permite recuperar o endereço desta variável (pode ser lido como "endereço de")

```
mediaPtr = &media;
```

- Operador de dereferência (*): quando utilizado sozinho, à esquerda de um ponteiro, retorna o conteúdo do ponteiro (pode ser lido como "conteúdo ou valor apontado por")

```
cout << "media=␣" << *mediaPtr << endl;
```

Exemplo: exemplo01.cpp

```
#include <iostream>

int main(){
  int x, *ptr;
  x = 5;
  ptr = &x;
  std::cout << "x␣␣␣␣=␣" << x << std::endl;
  std::cout << "*ptr␣=␣" << *ptr << std::endl;
  *ptr = 10;
  std::cout << "x␣␣␣␣=␣" << x << std::endl;
  std::cout << "*ptr␣=␣" << *ptr << std::endl;
  return 0;
}
```

Inicialização de ponteiros

- É uma boa prática de programação em C/C++ sempre inicializar tanto variáveis quanto ponteiros
- Em Java, o compilador inicializa as variáveis automaticamente com 0 (zero), mas em C/C++ isso NÃO ocorre
- Em C++, para ponteiros pode-se usar `nullptr`:

```
int *dados = nullptr;
```

- Para uma discussão sobre o uso de 0 ou NULL em C++, veja: `http://www.cplusplus.com/forum/beginner/5604/`

Impressão de ponteiros

- Em C++, ponteiros podem ser impressos usando cout normalmente:

```
#include <iostream>

int main() {
  int x;
  int *ptr;
  ptr = &x;
  std::cout << "O endereco de x é: " << ptr << std::endl;
  return 0;
}
```

Vetores

- Em C/C++, "um vetor é um ponteiro para determinado tipo de dado":

```
#include <iostream>

int main() {
  int vetor[10]= {1,2,3,4,5,6,7,8,9,10};
  std::cout << vetor << std::endl;
  std::cout << &vetor[0] << std::endl;
  std::cout << *vetor << std::endl;
  return 0;
}
```

Aritmética de Ponteiros

- É possível realizar operações sobre ponteiros
- Somar um valor inteiro a um ponteiro de determinado tipo significa "deslocar" a referência
- O deslocamento será igual a `sizeof(tipo)*valorInteiro`

```
#include <iostream>
int main() {
  int numeros[10]= {1,2,3,4,5,6,7,8,9,0};
  int *pnum = numeros;
  std::cout << *(pnum + 5) << std::endl;     // 6
  std::cout << pnum[5] << std::endl;         // 6
  std::cout << *(numeros + 5) << std::endl; // 6
  std::cout << numeros[5] << std::endl;      // 6
  return 0;
}
```

Ponteiros de Objetos

```
#include <iostream>

using namespace std;

class Pessoa {
  private:
    string nome;
  public:
    Pessoa(string n="") { nome = n; }
    string obtemNome()  { return nome; }
};

int main() {
  Pessoa *fulano = new Pessoa("Fulano de Tal");
  cout << fulano->obtemNome() << endl;
  delete fulano;
  return 0;
}
```

Alocação Dinâmica em C++

Alocação Dinâmica (Revisão)

- Alocação Estática

```
int num;
char v[10];
```

- Alocação Dinâmica: em C++ é feita com os operadores `new` (para alocar) e `delete` (para desalocar)

```
int *ptr_i = new int;
MinhaClasse *meuObjeto = new MinhaClasse;
// ...
delete meuObjeto;
delete ptr_i;
```

Alocação Dinâmica de Vetores

- No exemplo anterior, foi alocado um espaço para armazenar apenas um dado
- Para alocar um vetor, basta especificar o tamanho desse vetor no momento da alocação
- O acesso é feito exatamente como se fosse um vetor

```
#include <iostream>
int main() {
  int numeros[10]= {1,2,3,4,5,6,7,8,9,0};
  int *pnum = numeros;
  std::cout << *(pnum + 5) << std::endl;     // 6
  std::cout << pnum[5] << std::endl;         // 6
  std::cout << *(numeros + 5) << std::endl; // 6
  std::cout << numeros[5] << std::endl;      // 6
  return 0;
}
```

Alocação Dinâmica de Matrizes

```
#include <iostream>
using namespace std;
int main(){
  int **m;
  const int nLinhas = 3, nColunas = 4;
  m = new int*[nLinhas];
  for (int i=0; i<nLinhas; i++)
      m[i] = new int[nColunas];
  for (int i=0; i<nLinhas; i++)
      for (int j=0; j<nColunas; j++)
          m[i][j] = i*nLinhas+j;
  for (int i=0; i<nLinhas; i++) {
      cout << *m[i] << " : ";
      for (int j=0; j<nColunas; j++)
          cout << m[i][j] << " , " ;
      cout << endl;
  }
  for (int i=0; i<nLinhas; i++)
      delete [] m[i];
  delete [] m;
  return 0;
}
```

Realocação de Memória

- Se for preciso aumentar o tamanho de um vetor alocado dinamicamente, pode-se usar: `realloc(ponteiro, novo_tamanho)`
- Exemplo:

```
// A area de memoria eh realocada, e todos os dados sao
// copiados da area antiga para a area nova
cadastro = (CPessoa *)realloc(cadastro, 20);
```

Lista de Exercícios

Exercício 1

1. Analise o programa abaixo, considerando que as variáveis globais a (`int`, 4 *bytes*), b (`int`, 4 *bytes*), `ptr` (ponteiro para `int`, 8 *bytes*) e v (vetor com 4 elementos `int`, 16 *bytes*) tenham sido alocadas nos endereços especificados nos respectivos comentários, ou seja, respectivamente, nos endereços `0xf180`, `0xf184`, `0xf188` e `0xf190`. A partir destas informações, mostre a saída do programa.

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

int a;      // &a = 0xf180
int b;      // &b = 0xf184
int *ptr;   // &c = 0xf188
int v[4];   // &v[0] = 0xf190 / &v[1] = 0xf194 / &v[2] = 0xf198 / &v[3] = 0xf19c

int main() {
  a=4; b=4;
  ptr = &v[3];
  cout << ptr << endl;
  *ptr = b;
  ptr = ptr-1;
  *ptr = 9;
  ptr[a-5] = 12;
  v[0] = *(ptr+1);
  cout << *ptr << endl;
  ptr = &v[0];
  ptr[1] = 3;
  for (int i=0; i<4; i++) {
      cout << *ptr << endl;
      ptr++;
  }
  return 0;
}
```

Exercício 2

2. Analise a classe do programa abaixo e crie os métodos necessários para o armazenamento dos dados digitados pelo usuário.

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

class VetorDinamico {
private:
  // Declarar variáveis de instância necessárias
public :                                // Metodos a serem criados:
  VetorDinamico(int t = 10) {}          // Aloca o vetor com um tamanho recebido como parametro
  ~VetorDinamico() {}                   // Desaloca o vetor
  void adiciona(int v) {}               // Coloca um dado no vetor, realocando o vetor se necessário
  int obtemTamanho() { return -1; }     // Retorna o tamanho atual do vetor
  int obtem(int i) { return -1; }       // Retorna a informação da posição i do vetor, ou -1 se i for inválido
};

int main () {
  VetorDinamico vetor(5);  int dado;
  do {
    cin >> dado;  if ( dado > 0 ) vetor.adiciona(dado);
  } while (dado > 0);
  for (int i =0; i < vetor.obtemTamanho(); ++i)  cout << vetor.obtem(i) << endl;
  return 0;
}
```

Exercícios 3-6

3. Escreva um programa em C++ que recebe um vetor de números inteiros. Esse vetor de 10 posições deve ser criado de forma dinâmica e inicializado de 1 a 10. Ao final, deve-se imprimir na tela usando aritmética de ponteiros.
4. Crie duas funções em C++, uma para imprimir um vetor de inteiros e outra para imprimir um vetor de valores reais (`double`). Ambas as funções devem receber um ponteiro para o vetor e o tamanho do vetor, e devem usar apenas a aritmética de ponteiros.
5. Escreva um programa em C++ que inicialize um vetor de 100 números reais (`double`) com valores aleatórios quaisquer, e depois calcule a média apenas dos números que se encontram nos índices pares (iniciando em 0) desse vetor. Utilize apenas aritmética de ponteiros na sua implementação.
6. Modifique o programa do exercício anterior de forma que ele receba o tipo de média a ser calculada como parâmetro da linha de comandos: se for passado "0", faça a média sobre os índices pares; se for passado "1", faça a média sobre os índices ímpares; e, caso seja outro número, faça a média de todos os números do vetor.

Créditos

Créditos

- Estas lâminas contêm trechos de materiais disponibilizados pelos professores Rafael Garibotti, Márcio Sarroglia Pinho, Marco Mangan, Matheus Trevisan e Edson Moreno.

Soluções

Exercício 1

```
Memória:                                Saída do programa:
========                                ==================
0xf180 | a    | 4                      0xf19c
0xf184 | b    | 4                      9
0xf188 | ptr  | 0xf190                 4
0xf190 | v[0] | 4                      3
0xf194 | v[1] | 3                      9
0xf198 | v[2] | 9                      4
0xf19c | v[3] | 4
```

Exercício 2: exercicio02.cpp

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

class VetorDinamico {
private:
  int tamanho, tamanhoMax, *vetor;
public :
  VetorDinamico(int t = 10) {
    tamanho = 0;  tamanhoMax = t;  vetor = new int[tamanhoMax];
    cout << "+ VetorDinamico(" << tamanhoMax << ") criado..." << endl;
  }
  ~VetorDinamico() {
    cout << "- VetorDinamico(" << tamanhoMax << ") destruído..." << endl;
    delete[] vetor;
  }
  void adiciona(int v) {
     if ( tamanho >= tamanhoMax ) {
        int *aux = new int[tamanhoMax + 10];
        for (int i=0; i<tamanho; ++i) aux[i] = vetor[i];
        delete[] vetor;
        vetor = aux;
        tamanhoMax += 10;
        cout << "! VetorDinamico(" << tamanhoMax << ") modificado..." << endl;
     }
     vetor[tamanho++] = v;
  }
  int obtemTamanho() { return tamanho; }
  int obtem(int i) { return (i<0 || i>=tamanho)?-1:vetor[i]; }
};

int main () {
  VetorDinamico vetor(5);  int dado;
  do {
    cin >> dado;  if ( dado > 0 ) vetor.adiciona(dado);
  } while (dado > 0);
  for (int i =0; i < vetor.obtemTamanho(); ++i)  cout << vetor.obtem(i) << endl;
  return 0;
}
```

Exercício 3: exercicio03.cpp

```cpp
#include <iostream>

using namespace std;

int main() {
  int  *v, *pv;

  v = new int[10];
  for (int i=0; i<10; i++)
      v[i] = i+1;

  pv = v;
  for (int i=0; i<10; i++)
      cout << *(pv + i) << endl;

  delete[] v;
  return 0;
}
```

Exercício 4: exercicio04.cpp

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

void imprimeVetorInt (int *v, int tam){
  for (int i=0; i<tam; i++)
      cout << *(v+i) << endl;
}

void imprimeVetorDouble (double *v, int tam){
  for (int i=0; i<tam; i++)
      cout << *(v+i) << endl;
}

int main() {
  int    vi[5] = {2, 3, 4, 5, 6};
  double vd[6] = {2.2, 3.3, 4.4, 5.5, 6.6, 7.7};

  imprimeVetorInt(vi, sizeof(vi)/sizeof(int));
  imprimeVetorDouble(vd, sizeof(vd)/sizeof(double));

  return 0;
}
```

Exercício 5: exercicio05.cpp

```cpp
#include <iostream>
#include <iomanip>
#include <ctime>

using namespace std;

int main() {
  double mediaIndPares, v[100], *ptr = v;

  srand(time(NULL));
  for (int i=0; i<100; ++i)
    *(ptr+i) = (rand() % 100)/10.0;

  mediaIndPares=0.0;
  for (int i=0; i<50; ++i, ptr+=2) // Atenção: ptr é modificado...
    mediaIndPares += *ptr;
  mediaIndPares /= 50.0;

  cout << fixed << setprecision(2) << mediaIndPares << endl;
  return 0;
}
```

Exercício 6: exercicio06.cpp

```cpp
#include <iostream>
#include <iomanip>
#include <ctime>

using namespace std;

int main(int argc, char *argv[]) {
  if (argc != 2) { cerr << "Quantidade␣inválida␣de␣argumentos!" << endl;  return 1; }
  int ini, inc, tam, param = atoi(argv[1]);
  double media, v[100], *ptr = v;

  srand(time(NULL));
  for (int i=0; i<100; ++i)
    *(ptr+i) = (rand() % 100)/10.0;

  if (param == 0)       { ini = 0; inc = 2; tam = 50; }   // PARES
  else if (param == 1) { ini = 1; inc = 2; tam = 50; }   // ÍMPARES
  else                  { ini = 0; inc = 1; tam = 100; }  // TODOS
  media=0.0;
  for (int i=ini; i<tam; ++i, ptr+=inc) // Atenção: ptr é modificado...
    media += *ptr;
  media /= tam;

  cout << fixed << setprecision(2) << media << endl;
  return 0;
}
```