Introdução a Programação

Ponteiros e Vetores

Tópicos da Aula

- Hoje aprenderemos que existe uma forte relação entre ponteiros e vetores
  - Associação entre ponteiros e vetores
  - Ponteiros constantes x Ponteiros variáveis
  - Passagem de ponteiros invés de vetores para funções
  - Comando sizeof

Associação entre Vetores e Ponteiros

- Considere a declaração:

```
int   v [10] ;
```

  - O símbolo v
    - Representa o vetor
    - É uma constante que representa seu endereço inicial
    - Aponta para o primeiro elemento do vetor

Ponteiros e Vetores (matrizes)

- Em C existe um relacionamento muito forte entre ponteiros e vetores
  - O compilador entende todo vetor e matriz como ponteiros, pois a maioria dos computadores é capaz de manipular ponteiros e não vetores
  - Qualquer operação que possa ser feita com índices de um vetor pode ser feita com ponteiros
  - O identificador de um vetor representa um endereço, ou seja, um ponteiro

Ponteiros e Vetores

- Como vimos, C permite aritmética de ponteiros
- Se tivermos a declaração

```
int   v [10] ;
```

- Podemos acessar elementos do vetor através de aritmética de ponteiros

`v + 0` ⟶ Aponta para (igual ao endereço do) primeiro elemento do vetor

`v + 1` ⟶ Aponta para o segundo elemento do vetor

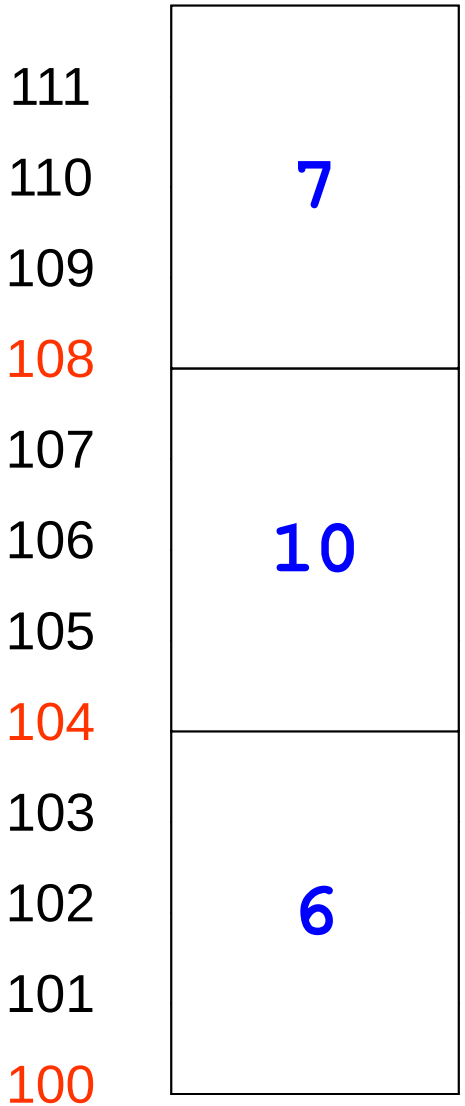
⋮

`v + 9` ⟶ Aponta para o último elemento do vetor

- Portanto: `&v[i]↔(v + i)` `v[i]↔ *(v + i)`

Representando Ponteiros e Vetores na Memória

Memória




```
int v[] = {6,10,7};
```

```
*(v + 2) ↔ v[2] ↔ 7
```

```
v + 2 ↔ &v[2] ↔ 108
```

```
*(v + 1) ↔ v[1] ↔ 10
```

```
v + 1 ↔ &v[1] ↔ 104
```

```
*v ↔ v[0] ↔ 6
```

```
v  ↔ &v[0] ↔ 100
```

Ponteiros e Vetores

Vetores podem ser tratados como ponteiros em C!

```
int a[10];
int *pa;
pa = &a[0];
pa = a;
```

→

```
*pa ↔ a[0] ↔ pa[0]
*(pa+i) ↔ a[i] ↔
pa[i] ↔ *(a+i)
a+i ↔ &a[i]
```

Expressões Equivalentes!

Usando Notação de Ponteiros para Vetores

Versão com Vetor

```
int main( ) {
    int nums[ ] = {1, 4, 8};
    int cont;
    for(cont=0; cont < 3; cont++) {
        printf("%d\n,nums[cont]);
    }
 }
```

Versão com Ponteiro

```
int main( ) {
    int nums[ ] = {1, 4, 8};
    int cont;
    for(cont=0; cont < 3; cont++) {
        printf("%d\n,*(nums + cont));
    }
 }
```

Ponteiros Constantes x Ponteiros Variáveis

```
int main( ) {
    int nums[ ] = {1, 4, 8};
    int cont;
    for(cont=0; cont < 3; cont++) {
       printf("%d\n,*(nums++));
    }
 }
```

Declaração de uma constante do tipo ponteiro para inteiros (ponteiro constante)

Errado!

Tenta incrementar endereço armazenado na constante **nums** e atualizar a constante com novo endereço

Ponteiros Constantes x Ponteiros Variáveis

```
int main( ) {
    int nums[ ] = {1, 4, 8};
    int* pnums = nums;
    int cont;
    for(cont=0; cont < 3; cont++) {
       printf("%d\n,*(pnums++));
    }
 }
```

Declaração de uma variável do tipo ponteiro para inteiros (ponteiro variável)

Certo!

Incrementa endereço armazenado na variável pnums e atualiza a variável com novo endereço

Ponteiros Constantes x Ponteiros Variáveis

```
int a[10];
int *pa;
pa = a;
```

Atribui a uma variável um novo endereço: CERTO!

```
int a[10];
int *pa;
a = pa;
```

Atribui a uma constante um novo endereço: ERRADO!

Passando Vetores como Argumentos para Funções

```
#include <stdio.h>
float  media(int n, float num[]){
  int i;
  float s = 0.0;
  for(i = 0; i < n; i++)
       s  = s +  num[i] ;
  return  s/n ;
}
int main(){
     float numeros[10] ;
     float med;
     int i ;
     for(i = 0; i < 10; i++)
          scanf ("%f", &numeros[i]) ;
     med  =  media(10, numeros ) ;
     ...
}
```

Parâmetro do tipo vetor de `float`

Endereço inicial do vetor é passado como argumento

Passando Ponteiros invés de Vetores como Argumentos para Funções

```
#include <stdio.h>
float  media(int n, float* num){
  int i;
  float s = 0.0;
  for(i = 0; i < n; i++)
       s  = s +  num[i] ;
  return  s/n ;
}
int main(){
     float numeros[10] ;
     float med;
     int i ;
     for(i = 0; i < 10; i++)
          scanf ("%f", &numeros[i]) ;
     med  =  media(10, numeros ) ;
     ...
}
```

Parâmetro do tipo ponteiro para float

Endereço inicial (ponteiro) do vetor é passado como argumento

Passando Ponteiros como Argumentos de Funções

- Considere a seguinte assinatura de função:

```
void incrementa(int n, int* v)
```

Pergunta: Parâmetro `v` é um ponteiro para um vetor de inteiros ou para uma variável do tipo inteiro?

Resposta 1: Não tem como saber

Resposta 2: É indiferente. Podemos considerar um ponteiro para uma variável do tipo inteiro como um ponteiro para um vetor com um só elemento

Comando *sizeof*

- Forma Geral:

```
sizeof(tipo) ou sizeof(variavel)
```

- Informa o número de bytes de um dado tipo ou variável em **tempo de compilação**
- Exemplo:

`int d = sizeof(float);` → d armazena o valor 4

Usando sizeof para Determinar Tamanho de Ponteiros e Vetores

Qual é o o numero de elementos?

3

```
int main() {
   int num[ ]={1,2,3};
   int numElementos = sizeof(num)/sizeof(int);
   printf ("Tamanho = %d\n", sizeof(num));
   printf ("Num elementos = %d\n", numElementos);
}
```

Qual é o o numero de elementos?

1

```
int main() {
   int num[ ]={1,2,3};
   int* num2 = num;
   int numElementos = sizeof(num2)/sizeof(int);
   printf ("Tamanho = %d\n", sizeof(num2));
   printf ("Num elementos = %d\n", numElementos);
}
```

Resumindo ...

- Relação entre ponteiros e vetores
- Ponteiros constantes x Ponteiros variáveis
- Passagem de ponteiros invés de vetores para funções
- Comando sizeof