Aula 10: Laboratório - Ponteiros (parte 2)

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

3Q-2017

Atividade em aula

Atividade em aula

1. Se `v` é um vetor, qual a diferença conceitual entre as expressões `v[70]` e `v+70`?

   [2 ponto]

   - v[70] está relacionado ao elemento na posição 71 do vetor v (conteúdo).
   - v+70 está relaciado ao endereço de memória do elemento na posição 71 do vetor v.

   A diferença é que o primeiro representa um elemento inteiro, já o segundo uma posição do vetor.

2. Seja `v` um vetor de inteiros. Suponha que cada inteiro ocupa 5 bytes no seu computador. Se o endereço de `v[0]` é `55000`, qual o valor da expressão `v+5`?

   [2 ponto]

   v+5 pode ser entendido como v+5*sizeof(int) = v+5*5 = 55025.

Atividade em aula

3. Escreva uma função `mm` que receba um vetor de inteiros `v[0..n-1]`, um inteiro `n` que indica o comprimento do vetor, e os endereços de duas variáveis inteiras, digamos `min` e `max`, e deposite nestas variáveis o valor de um elemento mínimo e o valor de um elemento máximo do vetor. Sua função **não** deve usar colchetes. [4 pontos]

```
// versão 1
void mm(int *v, int n, int *min, int *max) {
    int i;
    *min = *v;
    *max = *v;

    for (i=1; i<n; i++) {
        if (*min > *(v+i))
            *min = *(v+i);
        if (*max < *(v+i))
            *max = *(v+i);
    }
}
```

Atividade em aula

3. Escreva uma função `mm` que receba um vetor de inteiros `v[0..n-1]`, um inteiro `n` que indica o comprimento do vetor, e os endereços de duas variáveis inteiras, digamos `min` e `max`, e deposite nestas variáveis o valor de um elemento mínimo e o valor de um elemento máximo do vetor. Sua função **não** deve usar colchetes. [4 pontos]

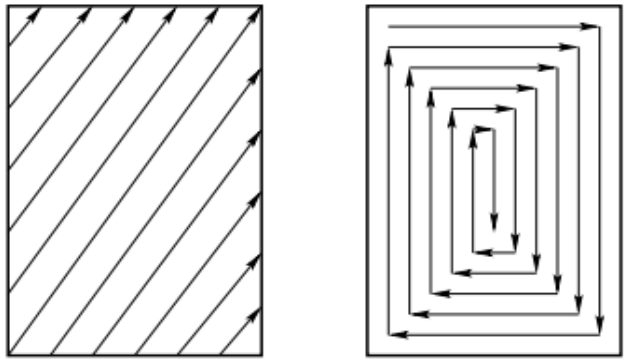
```
// versão 2
void mm2(int *v, int n, int *min, int *max) {
    int i;
    *min = *v;
    *max = *v;

    for (i=0; i<n; i++) {
        if (*min>*v)
            *min = *v;
        if (*max<*v)
            *max = *v;
        v++;
    }
}
```

Desafio 2

Desafio 2: 0.5 na MF

Para quem preferir incrementar sua nota final:

- Crie 2 funções que permitam percorrer uma matriz bidimensional seguindo os seguintes formatos.



- Seu programa não deve impor limitações sobre o número de linhas, nem colunas. **Não use colchetes. Use ponteiros.**
- Apresentação livre de exemplos (quanto mais completo melhor).
- Seu programa deve ser claro e bem indentado.

Desafio 2: 0.5 na MF

Envio:

- Tidia4 (aba Atividades)
- Data: 29/out, 23h50
- Trabalho **individual e opcional** (será utilizado um programa para detecção de plágio)
- Nome do arquivo: `desafio2_RA12345678.c`

Lista 4

Disciplina: Programação Estruturada
Turmas: A1 e A2 – Noturno

Prof. Dr. Jesús P. Mena-Chalco
Assistente Docente: Rafael J. P. Damaceno

Lista 4 - Deadline: 25/10 (23h50)

Nesta lista será trabalhado o tópico **Recursão** e **Ponteiros**. Usaremos a Plataforma URI para a avaliação de todos os problemas da lista: https://www.urionlinejudge.com.br.

1. Problema 1059. Números pares. Crie uma função recursiva para resolver este problema.
2. Problema 1073. Use ponteiro pelo menos uma vez no código.
3. Problema 1074. Par ou ímpar. Use ponteiro pelo menos uma vez no código.
4. Problema 1157. Divisores I. Escreva uma função recursiva para resolver este problema.
5. Problema 1164. Divisor perfeito. Use ponteiro pelo menos uma vez no código. Sugestão: use a função recursiva criada no Problema 1157.
6. Problema 1180. Menor e posição. Use ponteiro pelo menos uma vez no código.
7. Problema 1179 (Opcional). Preenchimento de Vetor IV. Use alocação dinâmica e aritmética de ponteiros para percorrer vetores.

Observações:

- **É obrigatório usar recursão e ponteiro nos casos indicados.**
- Será utilizado um programa especializado para detecção de plágio em todas as submissões.

Lista 4

```c
#include <stdio.h>

void pares(int n){
    if (n == 0){
        return;
    } else {
        if ((102 - n) % 2 == 0){
            printf("%d\n", 102 - n);
            pares(n - 2);
        }
    }
}

int main() {
    pares(100);
    return 0;
}
```

1059

Lista 4

```c
#include <stdio.h>

void pares(int n) {
    if (n<=100) {
        printf("%d\n", n);
        pares(n+2);
    }
}

int main() {
    pares(2);
    return 0;
}
```

1059

Lista 4

```c
#include <stdio.h>

void eh_div(int i, int j){
    if (i == j){
        printf("%d\n", j);
        return;
    } else {
        if (j % i == 0){
            printf("%d\n", i);
        }
        return eh_div(i + 1, j);
    }
}

int main() {

    int n;
    scanf("%d", &n);
    eh_div(1, n);

    return 0;
}
```

1157

Lista 4

```c
#include <stdio.h>

void eh_div(int n, int i) {
  if (i<=n) {
    if (n%i==0)
      printf("%d\n", i);
    eh_div(n, i+1);
  }
}

int main() {
  int n;
  scanf("%d", &n);

  eh_div(n, 1);

  return 0;
}
```

1157

Prática

exemploPonteiro.c

```c
#include<stdio.h>

int main() {
    int x;
    int i = 100;

    int *p;          /* p é um ponteiro para um inteiro */
    p = &i;          /* p aponta para i*/

    x = *p+900;      /* o mesmo que x = i+900 */

    printf("O valor de i   : %d\n", i);
    printf("O endereco de i: %p\n", &i);
    printf("O valor de p   : %p\n", p);
    printf("O valor de x   : %d\n", x);

}
```

```
O valor de i   : 100
O endereco de i: 0x7ffd987e5930
O valor de p   : 0x7ffd987e5930
O valor de x   : 1000
```

```
%d %i      Decimal signed integer.
%o         Octal integer.
%x %X      Hex integer.
%u         Unsigned integer.
%c         Character.
%s         String. See below.
%f         double
%e %E      double.
%g %G      double.
%p         pointer.
```

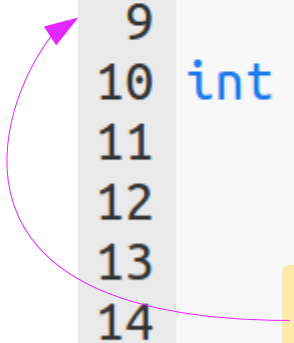
exemploPonteiro2.c

```c
#include<stdio.h>

int main() {
    int i;
    i = 100;

    printf("%d\n", *&i);

}
```

Qual é o resultado para `&*i` ?

exemploPonteiro3.c

```c
#include<stdio.h>

void troca(int *i, int *j) {
    int temp;
    temp = *i;
    *i = *j;
    *j = temp;
}

int main() {
    int a=1;
    int b=10;

    troca(&a,&b);

    printf("\ta=%d\n\tb=%d", a, b);
}
```



```
a=10
b=1
```

Por que o código abaixo está errado?

```
void troca(int *i, int *j) {
    int *temp;
    *temp = *i;
    *i = *j;
    *j = *temp;
}
```

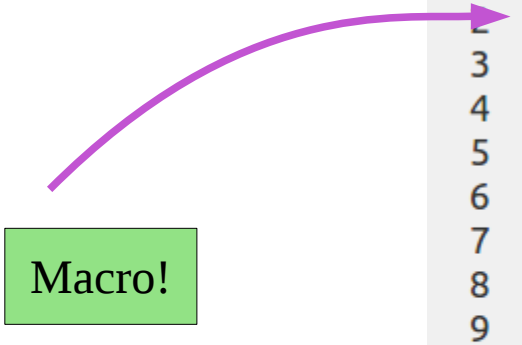
vetor1.c

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *v;
    int n, i;

    scanf( "%d", &n);

    v = (int *) malloc( n*sizeof(int) );

    for (i=0; i<n; i++)
        scanf( "%d", &v[i]);

    for (i=n; i>0; i--)
        printf( "%d ", v[i-1]);

    free(v);
}
```

Os ponteiros facilitam a **alocação dinâmica de memória**

```
6
1
2
3
4
5
6
6 5 4 3 2 1
```

vetor2.c




```c
#include <stdio.h>
#define max 100

void funcaoX(int M[max][max]) {
    M[2][3] = 4;
}

int main() {
    int A[max][max];
    A[2][3] = 5;

    funcaoX(A);

    printf("%d", A[2][3]);
}
```

```
4
```

vetor3.c

Escreva um programa que leia um número inteiro positivo **n** seguido de **n** números inteiros e imprima esses **n** números em ordem invertida.

Por exemplo, ao receber

```
5
22 33 44 55 66
```

o seu programa deve imprimir

```
66 55 44 33 22
```

- **Seu programa não deve impor limitações sobre o valor de n**
- **Seu programa não deve usar colchetes.**

vetor3.c

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n, i;
    scanf("%d", &n);
    int *p = (int *) malloc(n*sizeof(int));

    for (i=0; i<n; i++)
        scanf("%d", p+i);

    for (i=n-1; i>=0; i--)
        printf("%d ", *(p+i));

    free(p);
}
```

```
5
11 22 33 44 55
55 44 33 22 11
```

Os ponteiros facilitam a alocação dinâmica de memória