Universidade de São Paulo – São Carlos
Instituto de Ciências Matemáticas e de Computação

Operações com Strings Introdução a Ponteiros e Funções

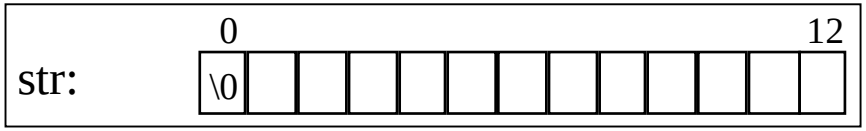
Profa Rosana Braga

Strings

- Strings são seqüências de caracteres adjacentes na memória.
- O caracter '\0' (= valor inteiro 0) indica o fim da seqüência

  Ex: `char str[13];`

  - define uma string de nome "str" e reserva para ela um espaço de 13 (12 + '\0') bytes na memória



Operações com Strings

- atribuição: não se pode atribuir uma string a outra string:

```
str = name;     /* erro */
```

- o header “string.h” contém uma série de funções para manipulação de strings:

strlen(str)	**retorna o tamanho de *str***
strcpy(dest, fonte)	**copia *fonte* em *dest***
strcat(dest, fonte)	**concatena *fonte* no fim de *dest***

Operações com Strings

- Ex:

```
char fonte[] = “Bom”;
char dest[] = “ dia!”;

strlen(fonte)      => retorna 3
strlen(dest)       => retorna 5
strcat(fonte, dest) => “Bom dia!”
strcpy(dest, fonte) => “Bom”
```

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

main()
{
  char fonte[] = "Bom";
  char dest[] = " dia!";
  printf("O tamanho da variavel fonte eh %d \n\n",strlen(fonte));
  printf("O tamanho da variavel dest eh %d \n\n",strlen(dest));
  printf("A concatenacao das duas variaveis eh %s
    \n\n",strcat(fonte,dest));
  printf("Apos concatenar temos fonte = %s e dest = %s \n\n",fonte,dest);
  strcpy(dest, fonte);
  printf("Agora copiei fonte em dest! Com isso, temos fonte = %s e dest =
    %s \n\n",fonte,dest);
  system("PAUSE");
}
```

Ponteiros

- um ponteiro é uma variável que contém um endereço
- Como declarar um ponteiro?

  "*" indica que a variável é um ponteiro

  tipo_dado *nome_ponteiro;

- Ex:

```
int  x;
int *px;      /* compilador sabe que px é ponteiro */
              /* px é um ponteiro para inteiro */
```

Ponteiros

- o operador “&” quando aplicado sobre uma variável retorna o seu endereço
- Ex:

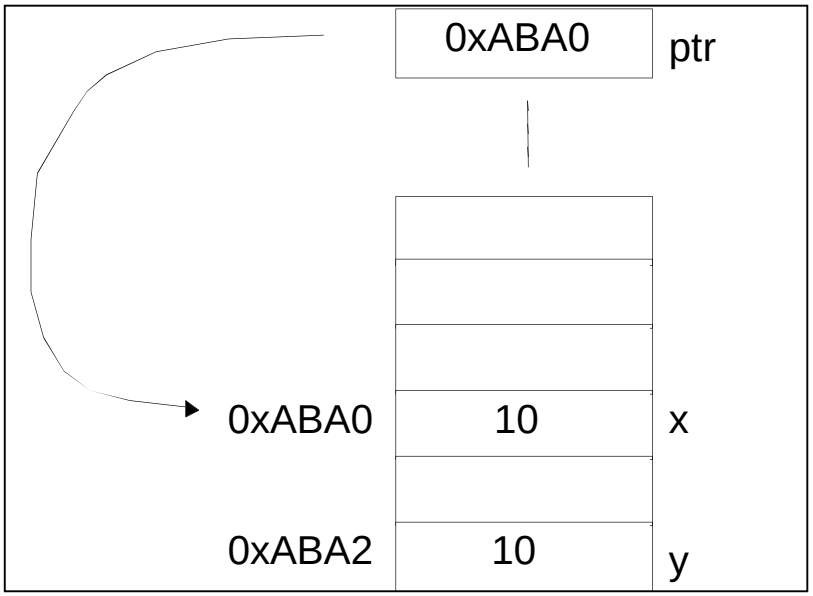
```
int x = 10, *pi;
pi = &x;
printf(“&x: %p pi: %p”, &x, pi);
```

=> &x: 0x03062fd8 pi: 0x03062fd8

Ponteiros

- o operador “*” quando aplicado sobre um ponteiro retorna o dado apontado
- Ex:

```
void main () {
    int *ptr;
    int x, y;
    x = 10;
    ptr = &x;
    y = *ptr;
}
```



```
#include <stdio.h>
#include <stdlib.h>
main()
{
   int *ptr, x, y;
   x = 10;
   ptr = &x;
   y = *ptr;
   printf("Variavel x = %d",x);
   printf("\n\nPonteiro ptr = %p",ptr);
   printf("\n\nConteudo do ponteiro *ptr = %d",*ptr);
   printf("\n\nEndereco de x &x = %p",&x);
   printf("\n\nVariavel y = %d",y);
   printf("\n\nEndereco de y &y = %p",&y);
   printf("\n\nEndereco do ponteiro &ptr = %p\n\n",&ptr);
   system("PAUSE");
}
```

Ponteiros

- ponteiros são variáveis tipadas:

  (int *) ≠ (float *) ≠ (char *)

- Ex:

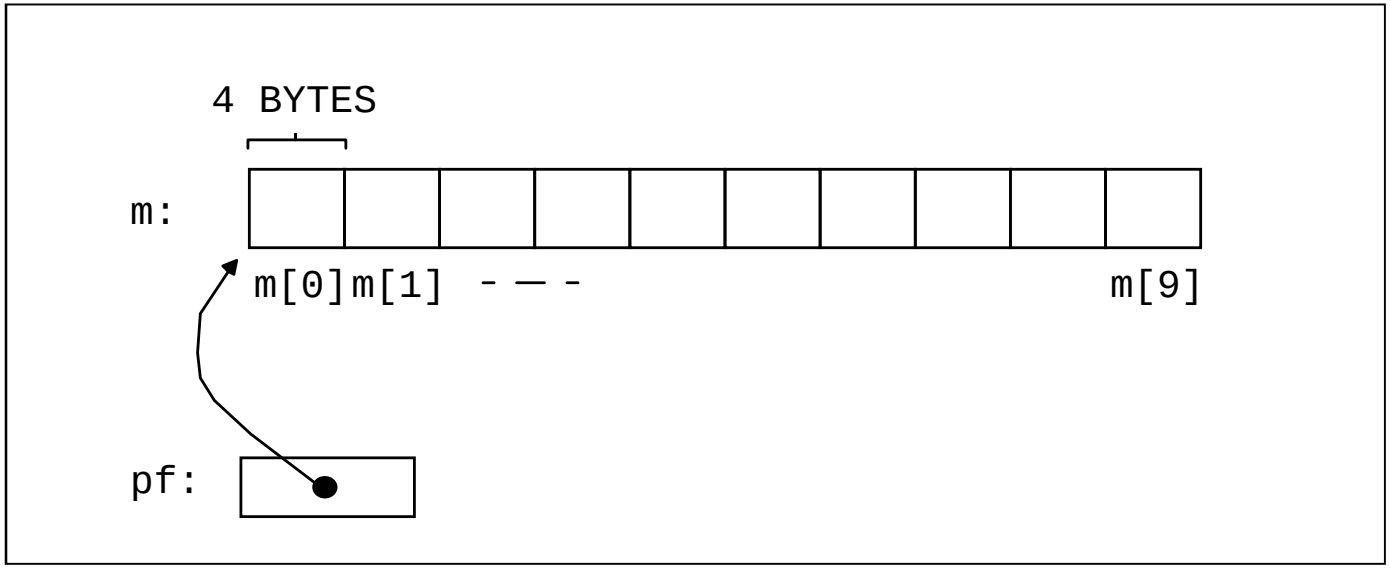
```
main() {
int *ip, x;
float *fp, z;
  ip = &x; /* OK */
  fp = &z; /* OK */
  ip = &z; /* erro */
  fp = &x; /* erro */
}
```

Ponteiro para Vetores

- Ex:

```
float m[10], *pf;
pf = m;
```

IMPORTANTE: pf = m; EQUIVALE A pf = &m[0];



```
#include <stdio.h>
#include <stdlib.h>
main()
{
   float m[4] = { 1.0, 3.0, 5.75, 2.345 };
            float *pf;
            int i;
            pf = &m[2];
            printf("Elemento 3 do vetor %f\n\n", *pf);

   /* percorrendo o vetor com indice tradicional */
   i=0;
   for (i=0; i<5; i++)
      printf(" %f ",m[i]);
   printf("\n\n\n");
   /* percorrendo o vetor com ponteiro*/
   pf = m;   /* equivale a pf = &m[0] */
   for (i=0; i<5; i++){
        printf(" %f ",*pf);
        pf++;
   }
   printf("\n\n\n");
   system("PAUSE");
}
```

Referenciando Elementos

- Pode-se utilizar ponteiros e colchetes:

```
float m[] = { 1.0, 3.0, 5.75, 2.345 };
float *pf;
pf = &m[2];
printf("%f", pf[0]);    /* ==> 5.75 */
```

- Note que o valor entre colchetes é o deslocamento a ser considerado a partir do endereço de referência

pf[n] => indica enésimo elemento a partir de pf

Função - Definição

- Agrupa um conjunto de comandos e associa a ele um nome
  - O uso deste nome é uma chamada da função
- Após sua execução, programa volta ao ponto do programa situado imediatamente após a chamada
  - A volta ao programa que chamou a função é chamada de retorno

Função

- A chamada de uma função pode passar informações (argumentos) para o processamento da função
  - Argumentos = lista de expressões
    - Lista pode ser vazia
    - Lista aparece entre parênteses após o nome da função
      - Ex.

```
int Soma (int x, int y) {
}
```

O Retorno da Função

- No seu retorno, uma função pode retornar resultados ao programa que a chamou

  return (resultados);

  - O valor da variável local *resultados* é passado de volta como o valor da função

- Valores de qualquer tipo podem ser retornados
  - Funções predicado: funções que retornam valores
  - Procedimentos: funções que não retornam valores

  Exemplo: void function (int x)

```
void main()
{
int n, fat=1;
     printf("Digite um numero inteiro: ");
     scanf("%d", &n);
     fat = fatorial(n);
     printf("\n\n\O fatorial de %d eh: %d\n\n", n, fat);
     system("pause");
}
int fatorial (int num){
   int f=1,cont;
   if (num > 0)   {
     cont = num;
     while (cont > 1){
         f = f * cont;
         cont--;
        }
     return f;
   }
   else
     return -1;
}
```

Exercício

- 1 – Modificar o programa anterior para que a função fatorial não retorne o valor do fatorial diretamente para o programa principal. Ao invés disso, ela deve receber um ponteiro para a variável que conterá o fatorial (no caso, o endereço de memória da variável fat) e trabalhar diretamente na variável. Ela recebe também como parâmetro o número o qual será calculado o fatorial.

Resposta do exercício 1

```
void main()
{
int n, fat=1;
     printf("Digite um numero inteiro: ");
     scanf("%d", &n);
     fatorial(&fat, n);
     printf("\n\nO fatorial de %d eh: %d\n\n", n, fat);
     system("pause");
}
void fatorial (int *pfat, int num){
   int cont;
   if (num > 0)   {
     cont = num;
     while (cont > 1){
         *pfat = *pfat * cont;
         cont--;
        }
   }
   else
     *pfat=-1;
}
```

Exercício

- 2 – Fazer um programa em C que leia 6 números, e armazene-os em um vetor.
  - Fazer uma função que recebe o ponteiro para o vetor e retorna a média aritmética dos elementos deste vetor.
  - O programa principal chama a função para calcular a média e mostra os números que estão abaixo da média.
- Dica: a função recebe também como parâmetro o tamanho do vetor.