Laboratório de Algoritmos I
Vetores, Matrizes e Strings

Vetor - conceito

- Vetor, array ou arranjo:
  - é uma coleção de variáveis do mesmo tipo que compartilham um mesmo nome.
  - uma maneira de armazenar vários dados num mesmo nome de variável através do uso de índices numéricos

- Vetores são úteis quando precisamos trabalhar com um grande número de elementos.

Vetor - declaração

- A declaração é parecida com aquela usada para variáveis convencionais.
- É necessário adicionar o tamanho do vetor à nossa declaração convencional:

```
int notas[5];
```

- Isso declara um vetor com cinco variáveis do tipo int (reserva o espaço para essas variáveis na memória)
- A forma genérica é:

```
tipo nome_da_variável[tamanho_do_vetor]
```

Vetor – referenciando um elemento

- Um vetor declara várias variáveis com um mesmo nome.
  - É necessário uma forma de acessar cada uma delas de forma independente.

- Referenciar o nome “notas” referencia o endereço do vetor. Colocando-se um índice, referencia-se a variável daquela posição do vetor.

```
notas[2] = 90;
```

- Atribui o valor 90 à posição 2 (2 é um índice inteiro) do vetor.

- Em C, os vetores começam na posição 0.
  - Um vetor de 10 elementos possui índices de 0 a 9.

Vetor – outros tipos

- Podemos declarar vetores usando qualquer tipo básico:

```
float notas[5];
double notas[5];
char letras[5];
```

Vetor – verificação de limites

- Em C não há verificação de limites do vetor.
- É responsabilidade do programador não acessar posições fora dos limites do vetor.

```
int x[5];
int i;
for (i = 0; i < 10; i++) {
     x[i] = i;
}
```

- O programa acima compila, mas não é correto!

Vetor – inicialização

- Abaixo um vetor que foi inicializado com o número de dias de cada mês do ano:

```
dmes[12] =
  {31,28,31,30,31,30,31,31,30,31,30,31}
```

ou

```
dmes[] =
  {31,28,31,30,31,30,31,31,30,31,30,31}
```

Matriz de duas dimensões

- É uma vetor de vetores.

```
int exemplo[2][2] = { {100,120} ,
  {200,240} };
```

Strings

- Um String é simplesmente uma cadeia de caracteres. Ou seja, usado para representar uma palavra, uma frase ou um texto qualquer.

- Em C um string é um vetor do tipo char, sempre terminada pelo caracter NULL (‘\0’).

Strings – Leitura

- Considerando que você tenha declarado, por exemplo, o string `char nome[80]`:

```
scanf("%s", nome); /*lê até o
  primeiro espaço*/
gets(nome); /*lê até a tecla
  ENTER*/
```

- Note que, com strings, não utilizamos o `&` no `scanf`.

Strings – Escrita

- Considerando que você tenha declarado, por exemplo, o string char nome[80]:

```
printf("%s\n", nome);
puts(nome); /* puts coloca um \n após a
     impressão, portanto os dois comandos
     acima fazem exatamente a mesma coisa
     */
```

Strings – Inicialização

```
char nome[] = {'A', 'n', 'a',
  '\0'};
```

ou

```
char nome[] = "Ana";
```

Matrizes de Strings

- Exemplo:

```
char diadasemana[7][14] =
  {“Domingo”, ”Segunda-feira”,
  ”Terca-feira”, ”Quarta-feira”,
  ”Quinta-feira”, ”Sexta-feira”,
  ”Sabado”};
```

Strings – função strlen

- Use `#include <string.h>`

- Retorna um inteiro com o tamanho do string (sem contar o caracter NULL do final).

- Exemplo:

```
int tamanho;
char nome[] = "Paulo";
tamanho = strlen(nome); /* tamanho receberá
  o valor 5 */
```

Strings – função strcat

- Use `#include <string.h>`
- Recebe dois strings e concatena o segundo no primeiro.
- Exemplo:

```
char nome1[80] = “Paulo”;
char nome2[80] = “ Souza”;
strcat (nome1, nome2); /* faz com que a
  string nome1 tenha o valor “Paulo Souza”.
  Não é verificado o tamanho de nome1.*/
```

Strings – função strcmp

- Use `#include <string.h>`
- Recebe dois strings e diz se são iguais.
- Exemplo:

```
char nome1[80] = "Paulo";
char nome2[80] = "Paulo";
strcmp (nome1, nome2); /* retorna 0 se são
  iguais. Fazendo-se nome1 == nome2 seriam
  comparados os endereços das matrizes. */
```

Strings – função strcpy

- Use `#include <string.h>`
- Recebe dois strings e copia o segundo para o primeiro.
- Exemplo:

```
char nome1[80] = “Paulo”;
char nome2[80] = “Souza”;
strcpy (nome1, nome2); /* nome1 passa a
  valer “Souza” */
```

Strings – funções strncat, strncmp e strncpy

- O mesmo que as funções anteriores, no entanto tem um terceiro parâmetro que diz quantos caracteres serão processados (concatenados, comparados, copiados, etc).

```
char nome1[80] = "Paulo";
char nome2[80] = "Souza";
strncpy (nome1, nome2, 3); /* nome1 passa a
  valer "Soulo" */
strncmp (nome1, nome2, 3); /* retorna 0, uma
  vez que as 3 primeiras letras de
  nome1(Soulo) são iguais às 3 primeiras de
  nome2(Souza)*/
strncat (nome1, nome2, 3); /*nome1 passa a
  valer "SouloSou" */
```