Revisão de C: Parte 1

Algoritmos e Estruturas de Dados 2
2017-1
Flavio Figueiredo (http://flaviovdf.github.io)

Boas práticas de C - Revisando AEDS1

Tópicos

- **Vetores e Strings**
- Passagem de parâmetros
- Entrada e saída
- Modularização
- Indentação
- Comentários
- Compilação e Debug

Alocação Estática de Memória

- Declaração de variáveis
- Espaço de memória suficiente é alocado
  - Espaço armazenado não é o mesmo para todos linguagens e/ou arquitetura
  - e.g., Java e Python podem alocar mais ou menos espaço

```
char c;     // 1 byte
short a;    // 2 bytes
int a;      // 4 bytes
long b;     // 8 bytes
float x;    // 4 bytes
double y;   // 8 bytes
```

Alocação Estática de Memória

- Ao fazer a alocação estática, apenas o espaço necessário na memória é reservado.

- O conteúdo de cada posição não é alterado
  - Uma variável apenas declarada pode conter qualquer coisa.

- **Inicializar as variáveis, atribuindo valores, antes do uso.**
  - **Inclusive vetores, matrizes e strings.**

```
int soma = 2;
```

- https://stackoverflow.com/questions/1262459/coding-standards-for-pure-c-not-c
- http://www.maultech.com/chrislott/resources/cstyle/indhill-cstyle.pdf

Vetores

- Aloca diversas variáveis
- Indexado por inteiros
- O valor entre chaves indica quantas vezes o espaço de 1 variável vai ser alocado
    - De forma simples, o número de elementos no vetor

```
int v[100];     // 100 * sizeof(int) = 400 bytes
long vl[100];   // 100 * sizeof(long) = 800 bytes
double z[100];  // 100 * sizeof(double) = 800 bytes
```

Vetores

- Indexados por números inteiros (referência)
- Tal referência indica a posição de memória onde buscar o valor
- Lembre-se de também iniciar os vetores

```
int num_elements = 1000;
float x[num_elements];

//Iniciando o vetor (boa prática)
for (int i = 0; i < 1000; i++) {
     x[i] = i * 3.0;
}

int valor = x[20];

// x[20] está na posição x+20*sizeof(float)
// Qual número está armazenado em valor?
```

Vetores

- **O compilador C não vai avisar se você está acessando uma região errada!**
- Um erro em tempo de execução vai ocorrer

```
float x[1000];
// ... código aqui inicializando etc...
float y = x[2000]; // não dá erro de compilação
```

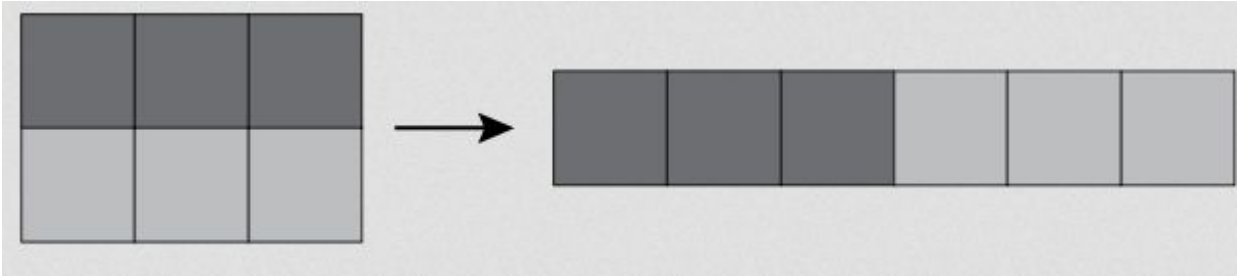
- Seu trabalho é garantir que tais erros não ocorram!
- Programe bem!
- Erro mais comum é o segmentation fault

Matrizes

- Vetores de mais de uma dimensão

```
int v[20][100];       // 20 * 100 * sizeof(int)
long vl[100][3];      // 100 * 3 * sizeof(long)
double z[100][100];   // 100 * 100 * sizeof(double)
```

- Por baixo a alocação é linear
  - Matriz 2 por 3 abaixo, cada linha é um tom de cinza (`int M[2][3];`)
  - Lado esquerda representa a memória



```
M[1][2]; // posição: M + (1*3+2)*sizeof(int)
```

Strings

- Representa texto escrito “abcdefghijklmnopqrstuvxz”
- Em C utilizamos um vetor de caracteres

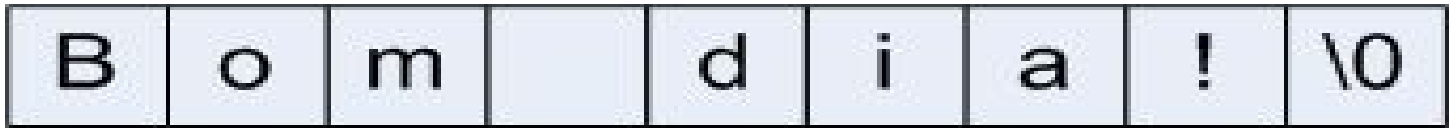
```
char nome[] = "Alexandre Silva";
//[A][l][e][x][a][n][d][e][ ][S][i][l][v][a][\0]
```

- Importante entender isto para saber como manipular texto
  - Atribuir e recuperar valores etc
- O caractere '\0' representa o fim da string (null em ASCII 0)
  - Útil quando é alocado mais espaço do que o tamanho da string
  - Sabemos onde termina o texto

Strings

- Para constantes utilizamos "

```
printf("%s", "Bom dia!");
```



- Novamente note a presença do \0
- Pergunta? Qual a diferença de:

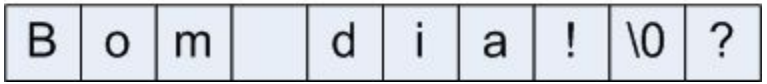
```
printf("%s\n", "Bom dia!");

printf("%s", "Bom dia!\n");
```

Strings

- Não é possível fazer atribuição direta para Strings
- Use strcopy ou strncopy
  - Diferença sutil entre as duas, em uma você pode passar o tamanho
  - No geral, pode utilizar strcopy, você já deve ter alocado o tamanho correto

```
char s[10];
strcpy(s, “Bom dia!”);
```



- Note o espaço no fim.

Strings

- As duas abordagens funcionam
- A segunda também pode ser utilizada para vetores de outros tipos

```
char[] nome = "Flavio";
char[] nome = {'F', 'l', 'a', 'v', 'i', 'o'};
```

- O nome da string representa o endereço de memória

```
char[] nome = 'Flavio';
printf("%s", nome + 3);
//Imprime vio
```

Funções de Strings

- strlen(st)
  - retorna o comprimento do string (com exceção do \0)
- strcat(s1, s2): concatena o s2 no s1
  - s1 tem que ter espaço suficiente alocado
- strcmp(s1, s2)
  - Comparação por ordem alfabética
  - retorna < 0 se s1 é menor que s2,
  - 0 se s1 é igual a s2,
  - e > 0 se s1 é maior que s2
  - **A comparação entre strings também tem que ser feita caractere a caractere, portanto não se pode usar s1==s2; isso só vai compara os endereços**

Exercício Poscomp 2009

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

int main (void) {
    char texto[] = "sem problemas";
    int i;
    for (i = 0; i < strlen(texto); i++)
    {
        if (texto[i] == ' ') break;
    }
    i++;
    for ( ; i < strlen(texto); i++)
         printf("%c", texto[i]);
    return 0;
}
```

- Qual vai ser a saída?

Strings

- strncat, strncmp e strncpy
  - Todas similares às respectivas anteriores
  - Especifica o número de caracteres

```
char str1[10];
char* str2 = "ABCDEFGHIJKLMNO";
strncpy(str1, str2, sizeof(str1));
printf("%s\n", str1);   // imprime "ABCDEFGHIJ"
```

Strings

- strtok: extrai tokens da string
- Vamos lembrar mais de AEDS1
- Explique o código ao lado
- Existem poucos trechos faltando (e.g., infile)

```
long int num; char linha[256]; char *p1 = NULL;
while (!feof(infile)) {
    fgets(linha, 256, infile);
    // delimitador: espaço ou fim de linha
    p1 = strtok(linha, " \n");
    while ((p1 != NULL) && (!feof(infile))) {
        num++; // Qual warning aconteceria aqui?
        fprintf(outfile, "%s\n", p1);
        p1 = strtok(NULL, " \n");
    }
}
printf("O arquivo tinha %ld palavras.\n", num);
fclose(infile);
```

Vetores de Strings

- Strings também são variáveis
- Podem ser declaradas dentro de vetores
- Dias da semana abaixo
- Nem todo dia tem 14 caracteres. Lembre-se do '\0'

```
char DiaSemana[7][14] =
    {"Domingo", "Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado"};
//...
printf("%s\n", DiaSemana[3]);
```

Vimos vetores e Strings
Continuamos na próxima aula
Devo passar o TP0

Trabalho Prático 0: Pontos Extra Revisão

- Computar 3 normas de matrizes
- Documentação na semana que vem
- Norma-1
  - Soma da coluna com a maior soma absoluta

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|,$$

- Norma-Inf
  - Soma da linha com maior soma absoluta

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|,$$

- Norma Frobenius
  - Raiz quadrada da soma dos valores elevado ao quadrado

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

Tópicos

- Vetores e Strings
- **Passagem de parâmetros**
- Entrada e saída
- Modularização
- Indentação
- Comentários
- Compilação e Debug