Programação Estruturada

Prof. Paulo Henrique Pisani

http://professor.ufabc.edu.br/~paulo.pisani/

novembro/2018

Constantes

- Constantes são valores fixos em um programa. São tratados como variáveis que não podem ter seu valor alterado.
- Há duas formas principais de declarar uma constante em C: com o pré-processador (**#define**) ou com a palavra-chave **const**.

Constantes com pré-processador

- Podemos incluir valores constantes com #define da seguinte forma:

```
#define <IDENTIFICADOR> <valor>
```

Constantes com pré-processador

- Exemplo:

```
#include <stdio.h>

#define MAX_N 50

int main() {
    char nome[MAX_N];
    fgets(nome, MAX_N, stdin);
    puts(nome);

    return 0;
}
```

Constantes com **const**

- Com a palavra-chave const, declaramos constantes assim:

```
const <tipo> <IDENTIFICADOR> = <valor>;
```

Constantes com **const**

- Exemplo:

```
#include <stdio.h>

const int MAX_N = 50;

int main() {
    char nome[MAX_N];
    fgets(nome, MAX_N, stdin);
    puts(nome);

    return 0;
}
```

Constantes com ponteiros

- Quando usamos constantes com ponteiros, podemos tornar o ponteiro constante ou a variável que ele aponta.
- Por exemplo:

```
const int *ptr;
```

Declara um ponteiro para um inteiro constante. Ou seja, o **valor do ponteiro pode** ser alterado, mas **a variável que ele aponta não**!

Constantes com ponteiros

- Quando usamos constantes com ponteiros, podemos tornar o ponteiro constante ou a variável que ele aponta.
- Por exemplo:

```
const int *ptr;

int v[3];
ptr = v;
ptr[2] = 40;
```

Constantes com ponteiros

- Esse tipo de constante pode ser usada quando queremos passar um ponteiro como parâmetro a uma função, mas não queremos que ela altere o conteúdo:

```
void imprimir_vetor(const int *v, int n) {
   int i;
   for (i = 0; i < n; i++)
      printf("%d ", v[i]);
   printf("\n");
}
```

Constantes com ponteiros

- Para o valor do ponteiro ser constante, declaramos da seguinte forma:

```
int v[5];
int * const ptr = v;
```

Declara um ponteiro constante para um inteiro. Ou seja, o **valor do ponteiro <u>não</u> pode** ser alterado, mas **a variável que ele aponta sim**!

Constantes com ponteiros

- Para o valor do ponteiro ser constante, declaramos da seguinte forma:

```
int v[5];
int * const ptr = v;

int v2[3];
ptr[2] = 507;
ptr = v2;
```

Constantes com ponteiros

- Para declarar um ponteiro constante que aponta para uma variável constante:

```
int v[5];
const int * const ptr = v;
```

Constantes com ponteiros

- Para declarar um ponteiro constante que aponta para uma variável constante:

```
int v[5];
const int * const ptr = v;

int v2[3];
ptr[2] = 507;
ptr = v2;
```

Bibliografia básica

- PINHEIRO, F. A. C. Elementos de programação em C. Porto Alegre, RS: Bookman, 2012.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. Lógica de programação: a construção de algoritmos e estruturas de dados. 3ª edição. São Paulo, SP: Prentice Hall, 2005.
- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: teoria e prática. 2ª edição. Rio de Janeiro, RJ: Campus, 2002.

Bibliografia complementar

- AGUILAR, L. J. Programação em C++: algoritmos, estruturas de dados e objetos. São Paulo, SP: McGraw-Hill, 2008.
- DROZDEK, A. Estrutura de dados e algoritmos em C++. São Paulo, SP: Cengage Learning, 2009.
- KNUTH D. E. The art of computer programming. Upper Saddle River, USA: Addison- Wesley, 2005.
- SEDGEWICK, R. Algorithms in C++: parts 1-4: fundamentals, data structures, sorting, searching. Reading, USA: Addison-Wesley, 1998.
- SZWARCFITER, J. L.; MARKENZON, L. Estruturas de dados e seus algoritmos. 3a edição. Rio de Janeiro, RJ: LTC, 1994.
- TEWNENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. Estruturas de dados usando C. São Paulo, SP: Pearson Makron Books, 1995.