Ponteiros para Funções

Módulo 8 Aula 3

Linguagem C, o Curso Definitivo WR Kits

Autor: Dr. Eng. Wagner Rambo

Quando um vetor é passado como parâmetro para uma função, na realidade não estamos lhe passando o vetor por completo, mas sim o endereço inicial do vetor. No Box 1, a função hipotética que recebe um vetor chamado vec, está na realidade recebendo &vec[0].

```
void func(int v[]);

main()
{
  int vec[10];

  func(vec);      /* estamos passando &vec[0] para func */

}
```

Box 1 - Ao invés do vetor completo, na realidade passamos o endereço do primeiro elemento.

Por este motivo, que a variável que recebe o endereço inicial deverá ser um ponteiro para o tipo de dados do vetor. A mesma função poderá ter o protótipo conforme o Box 2.

```
void func(int *v);
```

Box 2 - Função que recebe um vetor como parâmetro.

Lembrando que quando falamos em vetores, podemos também dizer matrizes unidimensionais e estas mesmas regras se aplicam para matrizes multidimensionais e *strings*.

Agora vamos analisar um recurso avançado pouco conhecido. Você sabia que é possível passar uma função como parâmetro para outra função? Em C isso é permitido, uma vez que uma função apresentará um ponto de entrada em seu código que nada mais é do que um endereço para o início desta função. Logo, se falamos sobre endereços, estamos falando intimamente com ponteiros. A função que receberá outra função como parâmetro – através de ponteiros – deverá ter o parâmetro compatível com o protótipo da função recebida. Veja no Box 3 um exemplo.

```
void receive(char *str, int (*func)(const char *));
```

Box 3 - Função que recebe outra função como parâmetro.

Neste caso, a função *receive* do Box 3 foi projetada para receber uma função que retorna um inteiro e recebe uma *string* como parâmetro. Vamos entender como isso funciona, apresentando uma função para verificar quantas letras a *string* contém, conforme Box 4.

```
int letters(const char *s)
{
  register int i=0, letter=0;

  while(s[i]!='\0')
  {
    if(isalpha(s[i]))
      letter++;

    i++;

  } /* end while */

  return letter;

} /* end letters */
```

Box 4 - Função para calcular quantas letras uma string contém.

Supondo que quisermos imprimir a quantidade de letras de uma *string* através da função *receive*, deveremos desenvolver esta função conforme o Box 5.

```
void receive(char *str, int (*func)(const char *))
{
   printf("Letras: %d\n",(*func)(str));

}
```

Box 5 - Corpo da função receive.

A chamada de *receive* na função principal será conforme o Box 6, utilizando o nome da função *letters* apenas, semelhante como fazemos quando passamos matrizes para funções.

```
main()
{
  char empresa[30] = “WR Kits”;

  receive(empresa,letters);     /* imprimirá “Letras: 6” */
}
```

Box 6 - Chamando a função receive.

Em algumas ocasiões específicas pode ser necessário passar funções arbitrárias para utilizá-las em seu código sem a necessidade de um *switch* com dezenas de *cases*. Bastando apenas utilizar a referência pelo índice de cada função.

Exercício proposto: a função *protocol* apresentada no Box 7 implementa um protocolo que verifica dois operandos. Sendo o primeiro operando maior, a função retorna um ‘H’ de *high*, do contrário, retorna um ‘L’ de *low*.

```
char protocol(int op1, int op2)
{
  char response;

  response = op1 > op2 ? 'H' : 'L';

  return response;

} /* end protocol */
```

Box 7 - Função de um protocolo hipotético, que indica com H ou com L a comparação de dois operandos.

Implemente uma função que irá receber dois operandos inteiros e a função *protocol* como parâmetro, interpretando o protocolo corretamente. A função apresentará a seguinte saída:

“Resultado do protocolo: H” para quando o primeiro operando for maior. Do contrário:

“Resultado do protocolo: L”

Bibliografia: DAMAS, Luís; Linguagem C, décima edição.

Disponível em: https://amzn.to/3nGdIbN