Variáveis Locais vs Globais

Módulo 6 Aula 3

Linguagem C, o Curso Definitivo WR Kits

Autor: Dr. Eng. Wagner Rambo

Variáveis Locais vs Globais

Uma variável local sempre existirá apenas no escopo da função em que ela foi declarada e poderemos utilizá-la apenas dentro da respectiva função. Uma variável global é aquela declarada fora do escopo de qualquer função, podendo ser acessada a qualquer momento em seu código.

As boas práticas sempre recomendam o uso de variáveis locais sempre que possível, para evitar possíveis erros como atualização indesejada de variáveis globais e outros efeitos colaterais em seu código.

As variáveis locais podem ter o mesmo nome, pois só existirão no escopo da função em que foram declaradas. As mesmas devem ser declaradas preferencialmente antes de qualquer outra instrução. No Box 1, apresentamos o exemplo para ficar mais claro.

```
void func();            /* protótipo das função */
int var_global;         /* declaração de variável global */

main()                  /* função principal */
{
  int var_local;        /* esta variável existirá apenas em main */

} /* end main*/

void func()             /* corpo da função */
{
  int var_local;        /* apesar de ter o mesmo nome, essa
                           variável existirá apenas em func */
} /* end func */
```

Box 1 - Exemplo de declaração de variáveis globais e locais.

O modificador volatile

É utilizado para informar ao compilador que o conteúdo de uma variável poderá ser alterado de forma implícita. Uma variável global pode ter seu endereço passado para uma rotina de RTC (*real time clock*) e ter seu conteúdo alterado sem qualquer sinal de atribuição evidente no programa. Existe um processo de otimização realizado por muitos compiladores que, quando não verificam atribuição alguma a uma variável no código, assumem que esta tem o conteúdo imutável. Neste contexto, o modificador volative poderá ser especialmente útil. No Box 2 um exemplo de declaração usando volatile.

```
volatile int time_val = 0;
```

Box 2 - Exemplo de declaração de variável volátil.

O especificador de classe de armazenamento static

Nas variáveis locais, *static* é utilizado quando desejamos preservar o conteúdo de uma variável entre uma e outra chamada de sua função. Se não utilizarmos, a variável é destruída sempre que a função se encerra. Em outras palavras, usando *static* a variável manterá o seu valor entre as chamadas da função. Isso é especialmente útil quando desejamos desenvolver funções independentes, que poderão ser reaproveitadas em outros códigos e projetos.

No caso das variáveis globais, o modificador *static* é utilizado para informar ao compilador que tal variável só será reconhecida naquele arquivo específico em que ela foi declarada.

```
static int value = 0;
```

Box 3 - Exemplo de declaração static.

O especificador de classe de armazenamento extern

Quando você desejar trabalhar com mais arquivos, uma possibilidade é declarar todas as suas variáveis globais no arquivo principal do projeto. Para poder acessá-las pelos demais arquivos do projeto, o especificador *extern* deve ser utilizado.

O Box 4 mostra o arquivo principal que conterá as variáveis globais.

```
int a=1, b=2, c=3;

main()
{
  results();    /* função presente no arquivo 2 */

}
```

Box 4 - Arquivo principal para exemplificar o uso de extern.

A função principal chama a função *result()*, que apresentará o resultado da soma das variáveis globais na tela. Inserindo um segundo arquivo no mesmo projeto, o mesmo poderá ser escrito conforme o Box 5.

```
extern a, b, c;

void results()
{
  int soma_tudo;


  soma_tudo = a+b+c;

  printf("%d\n", soma_tudo);

}
```

Box 5 - Segundo arquivo que contém a função e utiliza as variáveis globais do projeto.

O especificador de classe de armazenamento register

Este especificador basicamente solicita ao compilador que o conteúdo da variável seja armazenado em um registrador da CPU ao invés da memória. Teoricamente, as operações com variáveis do tipo register serão realizadas de forma muito mais rápida, uma vez que dispensaria a necessidade de acessar a memória. Isso também faz com que este tipo de variável não contenha um endereço, não podendo ser acessado através do operador &. O padrão C ANSI no entanto, apenas determina que o acesso à variável register seja realizado da forma mais rápida possível. O especificador register só pode ser aplicado em variáveis locais e parâmetros formais de funções. Um uso comum é na declaração de variáveis de iteração para laços *for*, conforme Box 6.

```
void repeat_8()
{
  register int i;

  for(i=0;i<8;i++)
   printf("Hello!\n");

}
```

Box 6 - Aplicação de register em uma determinada função.

O especificador de classe de armazenamento auto

Em C podemos utilizar o especificador *auto* para declarar variáveis locais, no entanto, todas as variáveis locais já assumem o especificador *auto* de forma implícita. Dizem que esta palavra reservada foi incluída em C por motivos de compatibilidade com sua antecessora B.

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

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