MAC 2166 – Introdução à Computação

POLI - PRIMEIRO SEMESTRE DE 2007

Material Didático

Prof. Ronaldo Fumio Hashimoto

ALGUNS DETALHES DA LINGUAGEM C

Introdução

Nesta aula vamos mostrar alguns detalhes da linguagem C.

Definição de Bloco

Os comandos dentro de uma repetição (comando `while`) são colocados entre chaves:

```
while (<condição>) {
  <comando_1>;
  <comando_2>;
  ...
  <comando_n>;
}
```

Todos os comandos que estiverem entre as chaves { e } estão dentro da repetição.

Agora, se dentro da repetição houver somente um comando, é possível omitir as chaves. Assim, no trecho de programa a seguir

```
while (<condição>)
  <comando_1>;

<comando_2>;
...
<comando_n>;
```

somente o `<comando_1>` está dentro da repetição. Os demais comandos estão fora! Note a correta indentação dos comandos.

O mesmo acontece com os comandos `if` e `if-else`. Compare os dois trechos de códigos:

```c
num = 1;
soma = 0;
while (num<n) {
  if (n % num == 0) {
    soma = soma + num;
  }
  num = num + 1;
}
if (soma == n) {
  printf ("numero perfeito\n");
}
else {
  printf ("nao eh perfeito\n");
}
```

```c
num = 1;
soma = 0;
while (num<n) {
  if (n % num == 0)
    soma = soma + num;
  num = num + 1;
}
if (soma == n)
  printf ("numero perfeito\n");
else
  printf ("nao eh perfeito\n");
```

Estes dois trechos são equivalentes, ou seja, produzem o mesmo resultado. No entanto, no trecho do lado esquerdo, todas chaves foram colocadas; enquanto que no trecho direito, algumas chaves foram omitidas.

Note que, uma vez que dentro do comando de repetição (`while`) temos dois comandos:

- um de seleção (`if (n % num == 0)`)
- e outro de atribuição (`num = num + 1`)

o uso das chaves é obrigatório.

Abreviaturas do Comando de Atribuição

Na linguagem C, é possível abreviar alguns comandos de atribuição.

Incremento e Decremento

É normal encontramos em problemas de MAC2166 situações em que se exigem que dentro de uma repetição devemos ter um incremento ou um decremento de uma variável. No exemplo anterior, tivemos a atribuição

```c
num = num + 1;
```

que significa aumentar de um o conteúdo da variável `num`. Na linguagem C é possível abreviar este comando fazendo:

```c
num++;
```

Exemplos de outras abreviaturas:

Descrição	Exemplo	Comando Abreviado
Incremento de um	`i = i + 1`	`i++`
Decremento de um	`n = n - 1`	`n--`
Incremento por uma variável	`soma = soma + num`	`soma += num`
Decremento por uma vairável	`soma = soma - num`	`soma -= num`
	`mult = mult * num`	`mult *= num`
	`divd = divd / num`	`divd /= num`
	`rest = rest % num`	`rest %= num`

Usando estas abreviaturas, o trecho de programa anterior poderia ser escrito como:

```
num = 1;
soma = 0;
while (num<n) {
  if (n % num == 0)
    soma = soma + num;
  num = num + 1;
}
if (soma == n)
  printf ("numero perfeito\n");
else
  printf ("nao eh perfeito\n");
```

```
num = 1;
soma = 0;
while (num<n) {
  if (n % num == 0)
    soma += num;
  num++;
}
if (soma == n)
  printf ("numero perfeito\n");
else
  printf ("nao eh perfeito\n");
```

Atribuição Múltipla

Em certas situações, podemos ter atribuições de um mesmo valor para várias variáveis. Por exemplo:

```
i = 0;
soma = 0;
count = 0;
```

É possível abreviar estas três linhas em uma única linha da seguinte forma:

```
i = soma = count = 0;
```

Atribuição na Declaração de Variáveis

Em alguns programas é necessário inicializar algumas variáveis. Por exemplo, no trecho de programa anterior, inicializar as variáveis `soma` e `num`.

É possível fazer estas inicializações destas variáveis na declaração das mesmas. Por exemplo:

```
# include <stdio.h>

int main () {
  int num, soma, n;

  printf ("Entre com n > 0: ");
  scanf ("%d", &n);

  num = 1;
  soma = 0;
  while (num<n) {
    if (n % num == 0)
      soma += num;
    num++;
  }
  if (soma == n)
    printf ("numero perfeito\n");
  else
    printf ("nao eh perfeito\n");

  return 0;
}
```

```
# include <stdio.h>

int main () {
  int num = 1, soma = 0, n;

  printf ("Entre com n > 0: ");
  scanf ("%d", &n);



  while (num<n) {
    if (n % num == 0)
      soma += num;
    num++;
  }
  if (soma == n)
    printf ("numero perfeito\n");
  else
    printf ("nao eh perfeito\n");

  return 0;
}
```

Note que no lado esquerdo, a inicialização das variáveis ocorre nos comandos de atribuição imediatamente anterior ao comando de repetição; enquanto que no lado direito, a inicialização das variáveis é feita na declaração das mesmas.

Definição de Constantes

Uma constante em um programa é um valor que não muda durante a sua execução. Diferente de uma variável, que durante a execução do programa pode assumir diferentes valores.

É possível definir constantes na linguagem C.

Vamos começar dando um exemplo. Suponha que no seu programa, você queira definir uma constante de nome `UM` que seja o número inteiro um e uma outra constante de nome `ZERO` que seja o número inteiro zero.

Aí, é possível fazer a seguinte atribuição:

```
num  = UM;
soma = ZERO;
```

Para definir uma constante, você deve fazer:

```
# define <NOME_DA_CONSTANTE> <VALOR>
```

Assim, para nosso exemplo, para definir `UM` e `ZERO` devemos fazer

```
# define UM 1
# define ZERO 0
```

Um programa completo com as constantes seria:

```
# include <stdio.h>

# define UM 1
# define ZERO 0

int main () {
  int num = UM, soma, n;

  printf ("Entre com n > 0: ");
  scanf ("%d", &n);

  soma = ZERO;
  while (num<n) {
    if (n % num == ZERO)
      soma += num;
    num++;
  }
  if (soma == n)
    printf ("numero perfeito\n");
  else
    printf ("nao eh perfeito\n");

  return 0;
}
```

Na verdade, definição e uso de constantes serão úteis quando estivermos trabalhando com vetores e matrizes.

Dúvidas

Dúvidas deste material pode ser enviadas para o “Fórum para assuntos específicos da turma WEB”.