Aulas 03 - Linguagem C - Estrutura básica

Conceito de variável

Neste capítulo abordaremos os primeiros elementos da linguagem C, que serão os conceitos iniciais que precisaremos para começar a escrever os primeiros programas. Começaremos pelas variáveis: quais os tipos existentes, regras para formação de nomes, etc.

Variáveis são regiões da memória às quais associamos um nome e que guardam um conteúdo, ou seja, são utilizadas para guardar as informações necessárias para a execução de um programa. Cada variável em um programa recebe um "apelido" ou nome associado ao endereço de memória que a contém.

Em C, quando definimos uma variável devemos definir também o seu tipo. O tipo da variável nos diz qual o tipo de dado que esta variável poderá armazenar.

Tipos pré-definidos

Em linguagem C existem vários tipos de variáveis pré-definidos, ou seja, tipos que são reconhecidos pelo compilador. Os tipos mais básicos e, portanto os mais usados são:

1. Tipo int

Armazena números inteiros e o tamanho do conjunto que pode ser representado normalmente depende da máquina em que o programa está rodando. Exemplo:

```
int ano;
int ano_nascimento;
```

2. Tipo float

As variáveis do tipo float também contêm valores numéricos, mas desta vez podendo apresentar posições decimais, ou seja, separadas através do caractere '.' (ponto decimal). Exemplo de valores reais que poderiam ser armazenados em variáveis float:

121.34 -19000.00 -0.12 3.1415926 34.00

3. Tipo char

Os caracteres são um tipo de dado: o char. Na linguagem C, também podemos usar um char para armazenar valores numéricos inteiros, além de usá-lo para armazenar caracteres de texto. Para indicar um caractere de texto usamos apóstrofes (aspas simples). Veja um exemplo:

```
char Ch;
Ch = 'D';
```

1.1. String

No C uma string é um vetor de caracteres terminado com um caractere nulo. O caractere nulo é um caractere com valor inteiro igual a zero (código ASCII igual a 0). O terminador nulo também pode ser escrito usando a convenção de barra invertida do C como sendo '\0'. Embora o assunto vetores seja discutido posteriormente, veremos aqui os fundamentos necessários para que possamos utilizar as strings. Para declarar uma string, podemos usar o seguinte formato geral:

```
char nome[tamanho];
```

Regras para formações de nomes de variáveis

Primeiramente com relação aos nomes das variáveis (e também das funções) em C elas podem ter qualquer nome se duas condições forem satisfeitas:

- O nome deve começar com uma letra ou *underscore* (muito chamado por *underline*) "_" - ex.: _nome;
- Os caracteres subsequentes devem ser letras, números ou *underscore* "_" – ex: nome_depto.

Além dessas duas condições acima, ainda há mais duas restrições:

- O nome de uma variável não pode ser igual a uma palavra reservada (palavra-chave), nem igual ao nome de uma função declarada pelo programador, ou pelas bibliotecas do C.
- Variáveis de até 32 caracteres são aceitas.

É bom sempre lembrar que C é uma linguagem "case sensitive" e, portanto deve-se prestar atenção às maiúsculas e minúsculas.

Dicas quanto aos nomes de variáveis:

- É uma prática tradicional do C usar letras minúsculas para nomes de variáveis e maiúsculas para nomes de constantes. Isto facilita na hora da leitura do código;
- Quando se escreve código usando nomes de variáveis em português, evitam-se possíveis conflitos com nomes de rotinas encontrados nas diversas bibliotecas, que são, em sua maioria absoluta, palavras em inglês.

Declaração das variáveis utilizadas

As variáveis no C devem ser declaradas antes de serem usadas. A forma geral da declaração de variáveis é:

```
tipo_da_variável lista_de_variáveis;
```

As variáveis da lista de variáveis terão o mesmo tipo e deverão ser separadas por "," vírgula.

Por exemplo, as declarações:

```
char ch, letra;
int cont;
float pi;
```

declaram duas variáveis do tipo char (ch e letra), uma variável do tipo int (cont) e um float (pi).

Outro exemplo: Você poderia encontrar em um determinado programa um trecho assim:

```
int idade, total;
float n1,n2,n3;
char inicial[8];
```

Neste trecho estão sendo declaradas duas variáveis inteiras (idade e total), três variáveis reais (n1, n2 e n3), e uma variável do tipo caractere (inicial).

Há três lugares nos quais podemos declarar variáveis. O primeiro é fora de todas as funções do programa. Estas variáveis são chamadas variáveis globais e podem ser usadas a partir de qualquer lugar no programa. Pode-se dizer que, como elas estão fora de todas as funções, todas as funções as veem. O segundo lugar no qual se pode

declarar variáveis é **no início** de um bloco de código. Estas variáveis são chamadas locais e só têm validade dentro do bloco no qual são declaradas, isto é, só a função à qual ela pertence sabe da existência desta variável, dentro do bloco no qual foram declaradas. O terceiro lugar onde se pode declarar variáveis é na **lista de parâmetros** de uma função. Mais uma vez, apesar destas variáveis receberem valores externos, estas variáveis são conhecidas apenas pela função onde são declaradas.

Se não for declarada alguma das variáveis utilizadas no programa, você será advertido com uma mensagem durante a compilação.

Atribuição para as variáveis

Nós podemos atribuir um conteúdo qualquer a uma variável (desde que seja do mesmo tipo definido), podemos imprimir este conteúdo na tela, utilizarmos este conteúdo em alguma expressão para cálculos, etc.

Para indicarmos uma atribuição sempre vamos utilizar o símbolo "=", que virá logo após o nome da variável à qual queremos atribuir, seguido ainda por uma constante ou uma expressão a ser calculada.

idade = 22	—>	A variável de nome **idade** está recebendo o número inteiro 22
n1 = 12.34	—>	Estamos atribuindo o valor real 12.34 à variável do mesmo tipo chamada n1
n3 = (n1+6)*2	—>	n3 está recebendo o resultado da expressão calculada (36.68)
n2 = n1	—>	n2 recebe o mesmo valor contido em n1 (12.34)
inicial = 'F'	—>	Guardamos o caratere 'F' na variável **inicial**, que foi definida como tipo caractere

Uma vez que o tipo das variáveis foi definido logo no início do programa, este tipo não poderá ser mudado e terá que ser respeitado até o final do programa. Qualquer tentativa de atribuição com um dado de um tipo inconsistente ao tipo definido para a variável ocasionará em uma mensagem de erro, detectada e acusada pelo compilador.

Exemplos de atribuições inválidas:

inicial = 'ITU'	—>	Dado tipo string atribuído à variável do tipo char
total = 'Z'	—>	Caractere de texto não pode ser guardado em variável do tipo int
idade = 35.4	—>	Valor real atribuído para uma variável do tipo int

Operadores aritméticos/prioridades

Os operadores aritméticos são usados para desenvolver operações matemáticas. A seguir apresentamos a lista dos operadores aritméticos do C:

Operador	Ação
+	Soma (inteira e ponto flutuante)
-	Subtração ou Troca de sinal (inteira e ponto flutuante)
*****	Multiplicação (inteira e ponto flutuante)
/	Divisão (inteira e ponto flutuante)
%	Resto de divisão (de inteiros)
++	Incremento (inteiro e ponto flutuante)
--	Decremento (inteiro e ponto flutuante)

O C possui operadores unários e binários. Os unários agem sobre uma variável apenas, modificando ou não o seu valor, e retornam o valor final da variável. Os binários usam duas variáveis e retornam um terceiro valor, sem alterar as variáveis originais. A soma é um operador binário, pois pega duas variáveis, soma seus valores, sem alterar as variáveis, e retorna esta soma. Outros operadores binários são os operadores - (subtração), *, / e %. O operador - como troca de sinal é um operador unário que não altera a variável sobre a qual é aplicado, pois ele retorna o valor da variável multiplicado por -1.

O operador / (divisão) quando aplicado a variáveis inteiras, nos fornece o resultado da divisão inteira; quando aplicado a variáveis em ponto flutuante nos fornece o resultado da divisão "real". O operador % fornece o resto da divisão de dois inteiros.

Assim seja o seguinte trecho de código:

```
int a = 17, b = 3;//variável a do tipo inteiro recebe valor 17
int x, y;
float z = 17. , z1, z2;


x = a / b;
y = a % b;
z1 = z / b;
z2 = a/b;//variável z2 do tipo float recebe a divisão de 2 números inteiros
```

ao final da execução destas linhas, os valores calculados seriam x = 5, y = 2, z1 = 5.666666 e z2 = 5.0 . Note que, na linha correspondente a z2, primeiramente é feita uma divisão inteira (pois os dois operandos são inteiros). Somente depois de efetuada a divisão é que o resultado é atribuído a uma variável float.

Os operadores de incremento e decremento são unários que alteram a variável sobre a qual estão aplicados. O que eles fazem é incrementar ou decrementar, a variável sobre a qual estão aplicados, de 1. Então,

```
x++;

x--;
```

são equivalentes a:

```
x = x + 1;

x = x – 1;
```

e também a:

```
x += 1;

x -= 1;
```

Estes operadores podem ser pré-fixados ou pós-fixados. A diferença é que quando são pré-fixados eles incrementam e retornam o valor da variável já incrementada. Quando são pós-fixados eles retornam o valor da variável sem o incremento e depois incrementam a variável. Então, em

```
x=23;
y=x++;
```

teremos, no final, y=23 e x=24. Em

```
x=23;
y=++x;
```

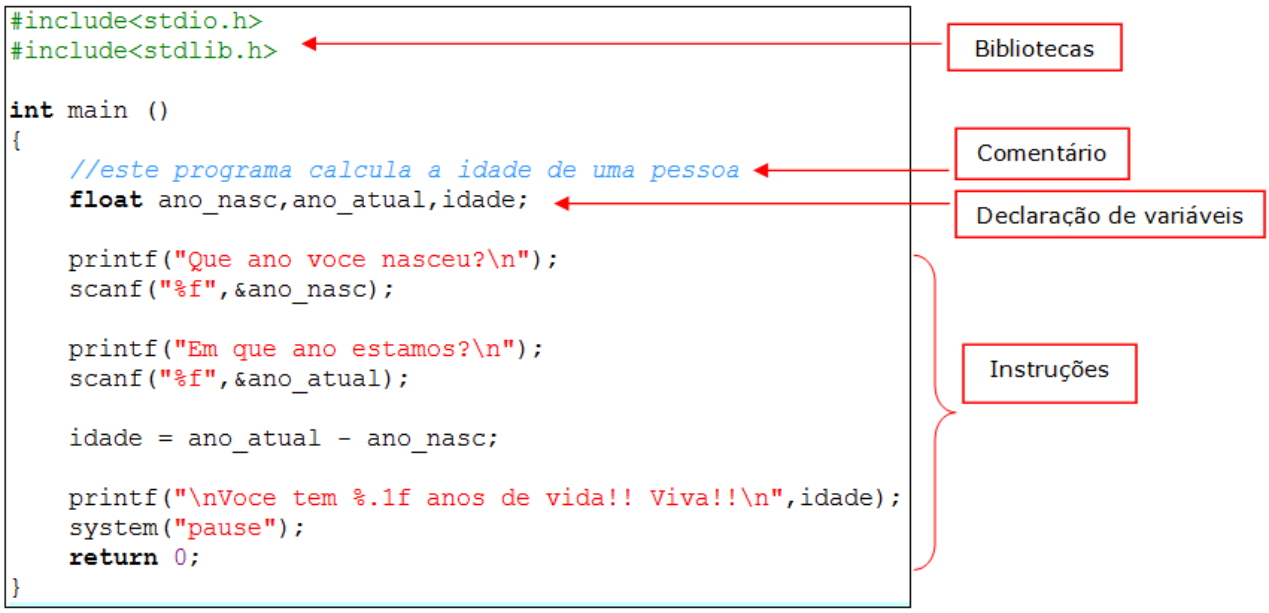
teremos, no final, y=24 e x=24.

Uma curiosidade: a linguagem de programação C++ tem este nome, pois ela seria um "incremento" da linguagem C padrão. A linguagem C++ é igual à linguagem C só que com extensões que permitem a programação orientada a objeto, o que é um recurso extra.

Estrutura básica de um programa em linguagem C

Todo programa em linguagem C consiste em uma ou mais funções devendo conter pelo menos a função primária "main ()". Segue-se uma estrutura padrão, regida por uma série de regras sintáticas, que tem por função organizar e estruturar as partes componentes do programa.

Esta estrutura pode ser esquematizada da seguinte maneira:



Vamos analisar o programa por partes

A linha **#include <stdio.h>** diz ao compilador que ele deve incluir o arquivo-cabeçalho (biblioteca) **stdio.h**. Nesta biblioteca existem declarações de funções úteis para entrada e saída de dados (std = standard, padrão em inglês; io = Input/Output, entrada e saída ==> stdio = Entrada e saída padronizadas). Toda vez que você quiser usar uma destas funções deve-se incluir este comando. O C possui diversas bibliotecas.

Quando fazemos um programa, uma boa ideia é usar comentários que ajudem a interpretar o funcionamento do mesmo. No caso acima temos um comentário: **//este programa calcula a idade de uma pessoa**. O compilador C desconsidera qualquer coisa que esteja começando com **//,** outra forma de fazer um comentário é utilizando no início **/*** e terminando com ***/**. Um comentário pode, inclusive, ter mais de uma linha.

A linha **int main()** indica que estamos definindo uma função de nome **main**. Todos os programas em C devem ter uma função **main**, pois é esta função que será chamada quando o programa for executado. O conteúdo da função é delimitado por chaves **{ }**. O código que estiver dentro das chaves será executado sequencialmente quando a função for chamada. A palavra int indica que esta função retorna um inteiro. O que significa este retorno será visto posteriormente, quando estudarmos um pouco mais detalhadamente as funções do C. A última linha do programa, **return 0**; indica o número inteiro que está sendo retornado pela função, no caso o número 0. O **\n** é uma constante chamada de *constante barra invertida*. No caso, o **n** é a constante barra invertida de "new line" e ele é interpretado como um comando de mudança de linha. É importante observar também que os *comandos* do C terminam com **;**

Sempre utilizaremos a identação (ou indentação) como técnica para salientar a divisão destes blocos. Outros recursos também são utilizados, como por exemplo, o uso de linhas em branco no meio do texto para separar melhor as diversas partes do programa.

Todos esses recursos além de melhorar a clareza e facilitar o entendimento do programa, também contribuirão para um visual melhor do programa facilitando sua interpretação por qualquer programador.

Aproveite e digite-o, compile e execute.

Atividades

1. Quais são os tipos de variáveis básicos pré-definidos do C?

2. Como pode ser esquematizada a estrutura básica de um programa em C?

3. Como você montaria as expressões para o computador?

   a) $\frac{5}{10} - \frac{3}{2} \times 1$

   b) $12^2 \times 5^2$

   c) 10% de 200

   d) $\frac{14 + 2 \times 4}{10}$

e) $\frac{(8 + 8) \times 4}{2}$

f) $\frac{((24 \times 2) - (12 + 5))}{2}$

g) { [(5 + 7) x 2] - [(3 x 3 + 1)] - 6 }