MC102

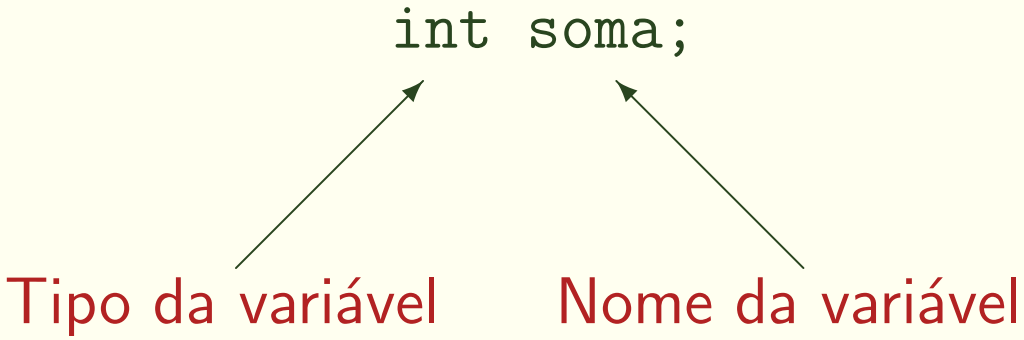
Algoritmos e programação de computadores

Aula 3: Variáveis

Variáveis

Variáveis são locais onde armazenamos valores na memória. Toda variável é caracterizada por um nome, que a identifica em um programa, e por um tipo, que determina o que pode ser armazenado naquela variável.

Declarando uma variável



Variáveis inteiras

- Variáveis utilizadas para armazenar valores inteiros, em formato binário.

  Ex: $13_{10} = 1101_2$

- `int`: Inteiro cujo comprimento depende do computador. É o inteiro mais utilizado. Em computadores *Pentium*, ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647.

Variáveis inteiras

- `unsigned int:` Inteiro cujo comprimento depende do computador e que armazena somente valores positivos. Em computadores *Pentium*, ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295.
- `long int:` Inteiro que ocupa 32 bits e pode armazenar valores de -2.147.483.648 a 2.147.483.647, independente do computador.
- `unsigned long int:` Inteiro que ocupa 32 bits e pode armazenar valores de 0 a 4.294.967.295, independente do computador.

Variáveis inteiras

- `short int`: Inteiro que ocupa 16 bits e pode armazenar valores de -32.768 a 32.767.
- `unsigned short int`: Inteiro que ocupa 16 bits e pode armazenar valores de 0 a 65.535.

Variáveis de tipo caracter

- Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos.
- São, na verdade, variáveis inteiras que armazenam um número associado ao símbolo. A principal tabela de símbolos utilizada pelos computadores é a tabela ASCII (*American Standard Code for Information Interchang*), mas existem outras (EBCDIC, Unicode, etc ..).

Variáveis de tipo caracter

- `char`: Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de -128 a 127.
- `unsigned char`: Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de 0 a 255.

Variáveis de tipo ponto flutuante

- Armazenam valores reais, da seguinte forma

$$(-1)^{sinal} \cdot mantissa \cdot 2^{expoente}$$

  Ex: $0.5 = (-1)^0 \cdot 1 \cdot 2^{-1}$

- Para o programador, funciona como se ele armazenasse números na forma decimal.
- Possui problemas de precisão (arredondamento).

Variáveis de tipo ponto flutuante

- `float`: Utiliza 32 bits, sendo 1 para o sinal, 8 para o expoente e 23 para a mantissa. Pode armazenar valores de $(+/-)10^{-38}$ a $(+/-)10^{38}$

- `double`: Utiliza 64 bits, sendo 1 para o sinal, 11 para o expoente e 52 para a mantissa. Pode armazenar valores de $(+/-)10^{-308}$ a $(+/-)10^{308}$

Obtendo o tamanho de um tipo

O comando sizeof(tipo) retorna o tamanho, em bytes, de um determinado tipo. (Um byte corresponde a 8 bits).

Ex: `printf ("%d", sizeof(int));`

escreve `4` na tela (*Pentium*).

Regras para nomes de variáveis em C

- **Deve** começar com uma letra (maíuscula ou minúscula) ou subcrito(_). **Nunca** pode começar com um número.
- Pode conter letras maíusculas, minúsculas, números e subscrito.
- Não pode-se utilizar

  ```
  { ( + - * / \ ; . , ?
  ```

  como parte do nome de uma variável.

Regras para nomes de variáveis em C

As seguintes palavras já tem um significado na linguagem C e por esse motivo não podem ser utilizadas como nome de variáveis:

`auto`	`double`	`int`	`struct`	`break`
`enum`	`register`	`typedef`	`char`	`extern`
`return`	`union`	`const`	`float`	`short`
`unsigned`	`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`	`do`
`if`	`static`	`while`		

Exercício: quais dos nomes a seguir são nomes corretos de variáveis? Se não forem nomes corretos, porque não são?

3ab

a3b

FIM

int


meu


A

n_a_o

papel-branco

a*

c++

*nova_variavel

Constantes

- Constantes são valores previamente determinados e que, por algum motivo, devem aparecer dentro de um programa (veremos adiante onde elas podem ser usadas).
- Assim como as variáveis, as constantes também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo `string`, que corresponde a uma sequência de caracteres.
- Exemplos de constantes: 85, 0.10, 'c', "Meu primeiro programa".

Constantes inteiras

- Um número na forma decimal, como escrito normalmente

  Ex: 10, 145, 1000000

- Um número na forma hexadecimal (base 16), precedido de 0x

  Ex: 0xA ($0xA_{16} = 10_2$), 0x100 ($0x100_{16} = 256_2$)

- Um número na forma octal (base 8), precedido de 0

  Ex: 010 ($0x10_8 = 8_2$)

Constantes do tipo de ponto flutuante

- Um número decimal. Para a linguagem C, um número só pode ser considerado um número decimal se tiver uma parte "não inteira", mesmo que essa parte não inteira tenha valor zero. Utilizamos o ponto para separarmos a parte inteira da parte "não inteira"

  Ex: 10.0, 5.2, 3569.22565845

Constantes do tipo ponto flutuante

- Um número inteiro ou decimal seguido da letra e e um expoente. Um número escrito dessa forma deve ser interpretado como:

$$numero \cdot 10^{expoente}$$

Ex: 2e2 ($2e2 = 2 \cdot 10^2 = 200.0$)

Constantes do tipo caracter

- Uma constante do tipo caracter é sempre representado por uma letra entre aspas simples.

  Ex: 'A'

- Toda constante do tipo caracter pode ser usada como uma constante do tipo inteiro. Nesse caso, o valor atribuído será o valor daquela letra na tabela ASCII.

Constantes do tipo string

- Uma constante do tipo é um texto entre aspas duplas

  Ex: "Meu primeiro programa"

Escrevendo o conteúdo de uma variável na tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando `printf`. Para isso, utilizamos um símbolo no texto para representar que aquele trecho deve ser substituído por uma variável e, no final, passamos uma lista de variáveis ou constantes, separadas por vírgula.

Escrevendo o conteúdo de uma variável na tela

Ex: printf ("A variável %s contém o valor %d","a", a);

imprime `A variável a contém o valor 10`

- Nesse caso, %s deve ser substituído por uma variável ou constante do tipo `string` enquanto %d deve ser substituído por uma variável ou constante do tipo inteiro.

Formatos inteiros

%d — Escreve um inteiro na tela sem formatação.

Ex: `printf ("%d", 10)`

imprime 10

Formatos inteiros

$\%< numero >\mathbf{d}$ — Escreve um inteiro na tela, preenchendo com espaços a esquerda para que ele ocupe pelo menos $< numero >$ casas na tela.

Ex: `printf ("%4d", 10)`

imprime $< espaco >< espaco >$10

Formatos inteiros

$\mathbf{\%0}< numero >\mathbf{d}$ — Escreve um inteiro na tela, preenchendo com zeros a esquerda para que ele ocupe pelo menos comprimento $< numero >$.

Ex: `printf ("%04d", 10)`

imprime `0010`

Formatos inteiros

$\%< numero_1 >\mathbf{.0}< numero_2 >\mathbf{d}$ — Escreve um inteiro na tela, preenchendo com espaços a esquerda para que ele ocupe pelo menos $< numero_1 >$ casas na tela e com zeros para que ele possua pelo menos comprimento $< numero_2 >$.

Ex: `printf ("%6.04d", 10)`

imprime $< espaco >< espaco >$0010

Formatos inteiros

- A letra d pode ser substituida pelas letras u e l, ou as duas, quando desejamos escrever variáveis do tipo unsigned ou long, respectivamente.

  Ex: `printf ("%d", 4000000000)`

  escreve -294967296 na tela, enquanto que

  `printf ("%u", 4000000000)`

  escreve 4000000000.

Formatos ponto flutuante

%f — Escreve um ponto flutuante na tela, sem formatação

Ex: `printf ("%f", 10.0)`

imprime `10.000000`

Formatos ponto flutuante

%e — Escreve um ponto flutuante na tela, em notação científica

Ex: `printf ("%e", 10.02545)`

imprime `1.002545e+01`

Formatos ponto flutuante

$\%< tamanho >.< decimais >\mathbf{f}$ — Escreve um ponto flutuante na tela, com tamanho $< tamanho >$ e $< decimais >$ casas decimais. Lembre-se que o ponto, utilizado para separar a parte inteira da decimal, também conta no tamanho.

Ex: `printf ("%6.2f", 10.0)`

imprime $< espaco >$`10.00`

Formatos inteiros

- A letra f pode ser substituida pelas letras lf, para escrever um `double` ao invés de um `float`

  Ex: `printf ("%6.2lf", 10.0)`

  imprime $< espaco >$`10.00`

Formato caracter

`%c` — Escreve uma letra.

Ex: `printf ("%c", 'A')`

imprime `a`

Note que `printf ("%c", 65)` também imprime a letra A.

Formato `string`

%s — Escreve uma string

Ex: `printf ("%s", "Meu primeiro programa")`

imprime `Meu primeiro programa`