MC-102 — Aula 03
Bits e Base Numérica. Variáveis, Atribuições e Estrutura Básica de um Programa

Instituto de Computação – Unicamp

Primeiro Semestre de 2012

Roteiro

Bit - menor unidade de informacão usada pelo computador

Bit

Um bit tem atribuicões lógicas 0 ou 1, onde cada um destes estados pode, internamente, ser representado por meios eletro-magnéticos (negativo/positivo, ligado/desligado, etc).
É por isso que é mais fácil armazenar dados em formato binário.

- Assim, todos os dados do computador são representados de forma binária.
- Mesmo os números são comumente representados na base 2, em vez da base 10, e suas operações são feitas na base 2.

Caracterer e Tabela ASCII

- Um conjunto de 8 bits é chamado de byte e pode ter até 256 configurações diferentes ($2^8 = 256$).
- O principal padrão usado para representar caracteres (`'a'`, `'b'`, `'c'`,..., `'A'`, `'B'`, `'C'`,..., `'!'`, `'@'`, `'#'`,...) é o padrão ASCII (*American Standard Code for Information Interchange*), usado na maioria dos computadores.
- Cada um destes caracteres é representado por um byte.

Caracterer e Tabela ASCII

A tabela apresenta o código binário e o valor decimal correspondente de alguns caracteres no padrão ASCII:

Caracter	Representação em ASCII	Valor na base decimal
⋮	⋮	⋮
	00100000	32
!	00100001	33
”	00100010	34
#	00100011	35
$	00100100	36
⋮	⋮	⋮
0	00110000	48
1	00110001	49
2	00110010	50
3	00110011	51
⋮	⋮	⋮
A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
⋮	⋮	⋮
a	01100001	97
b	01100010	98
c	01100011	99
d	01100100	100
⋮	⋮	⋮

Caracterer e Tabela ASCII

Caracter	Representação em ASCII	Valor na base decimal
⋮	⋮	⋮
	00100000	32
!	00100001	33
”	00100010	34
#	00100011	35
$	00100100	36
⋮	⋮	⋮
0	00110000	48
1	00110001	49
2	00110010	50
3	00110011	51
⋮	⋮	⋮
A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
⋮	⋮	⋮
a	01100001	97
b	01100010	98
c	01100011	99
d	01100100	100
⋮	⋮	⋮

Sistema de numeração decimal e binário

- Sistema de numeração decimal é um sistema de numeração posicional em que todas as quantidades se representam com base em dez números (0, 1, 2, 3, 4, 5, 6, 7, 8 e 9).
- O sistema binário ou base 2, é um sistema em que todas as quantidades se representam com base em dois números (0 e 1).

Sistema de numeração decimal e binário

O número 4027 na base decimal representa a quantidade:

$$4*10^3 + 0*10^2 + 2*10^1 + 7*10^0;$$
$$4000 + 0 + 20 + 7 = 4027$$

O número 100110_b no sistema binário representa a quantidade:

$$1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$$
$$32 + 0 + 0 + 4 + 2 + 0 = 38 \texttt{ na base decimal}$$

Sistema de numeração decimal e binário

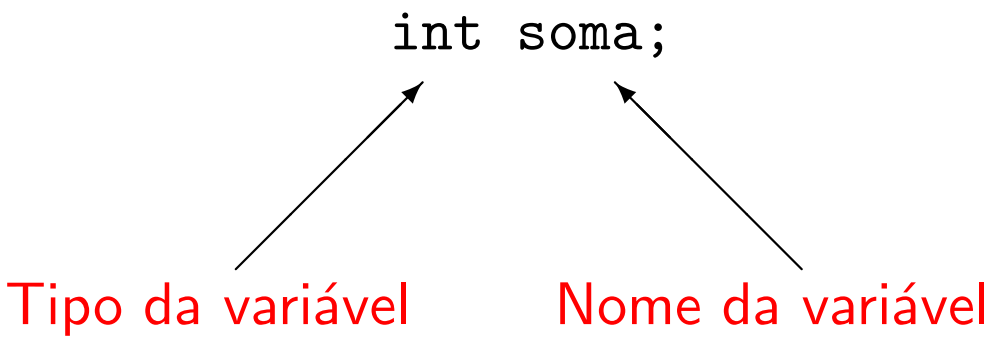
- Dado um número no sistema decimal de que maneira poderíamos escrevê-lo no sistema binário?
- Quantos números na base decimal é possível representar com 8 bits?

Variáveis

Definição

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



Declarando uma variável

Declara-se da seguinte forma: **Tipo_Variável Nome_Variável;**

Exemplos corretos:

- int soma;
- float preco_abacaxi;
- char resposta;

Exemplos incorretos:

- soma int;
- float preco_abacaxi

Variáveis inteiras

Variáveis utilizadas para armazenar valores inteiros.
Ex: $13_{10} = 1101_2$

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

Variáveis inteiras

- `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.
- `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 inteiras

Exemplos de declaração de variáveis inteiras:

- int numVoltas;
- int ano;
- unsigned int quantidadeChapeus;

Exemplos Inválidos:

- int int numVoltas;
- unsgned int ano;

Variáveis inteiras

Você pode declarar várias variáveis de um mesmo tipo. Basta separar as variáveis por vírgula:

- int numVoltas , ano;
- unsigned int a, b, c, d;

Variáveis de tipo caracter

Variáveis utilizadas para armazenar letras e outros símbolos existentes em textos. OBS: Guarda apenas um caracter.

- 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*).
- `char:` Armazena um símbolo (no caso, o inteiro correspondente). Seu valor pode ir de -128 a 127.
- Exemplos de declaração:
  - char umaLetra;
  - char YOuN;

Variáveis de tipo ponto flutuante

Armazenam valores reais, da seguinte forma

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

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

- Possuem problemas de precisão pois há uma quantidade limitada de memória para armazenar um número real.
- `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 $(+/-)10^{308}$

Variáveis de tipo ponto flutuante

Exemplos de declaração de variáveis de tipo ponto flutuante.

- float salario;
- float resultado, cotacaoDolar;
- double a, b, c;

O endereço de uma variável

- Toda variável tem um endereço de memória associado a ela. Esse endereço é o local onde essa variável é armazenada no sistema (como se fosse o endereço de uma casa, o local onde as pessoas "são armazenadas").
- Normalmente, o endereço das variáveis não é conhecido quando o programa é escrito.
- O endereço de uma variável é dependente do sistema computacional e também da implementação do compilador C que está sendo usado.
- O endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

Variáveis que guardam endereços

- Armazenam o endereço de outras variáveis.
- Para cada tipo de dados, existe um tipo para guardar o seu endereço, indicado por * antes do nome da variável.
- `int *endereço`: Endereço de uma variável inteira.
- `float *endereço`: Endereço de uma variável de ponto flutuante.
- `char *endereço`: Endereço de uma variável de caracter.
- Estas variáveis são chamadas apontadores.

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).

Exemplo

```
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:

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

- Letras maiúsculas e minúsculas são diferentes:

```
int c;
int C;
```

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`		

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', "Hello, world!"
```

Constantes

- Uma **constante inteira** é um número na forma decimal, como escrito normalmente
  Ex: `10`, `145`, `1000000`
- Uma **constante ponto flutuante** é um número real, onde a parte fracionário vem depois de um ponto
  Ex: `2.3456`, `32132131.5`, `5.0`

Constantes do tipo caracter e string

- 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.
- Uma constante do **tipo string** é um texto entre aspas duplas
  Ex: `"Hello, world!"`

Comando de Atribuição

Definição

O comando de atribuição serve para atribuir valores para variáveis.

- O comando de atribuição em C é o sinal =
- A sintaxe do uso do comando é:

  variável = valor ;

- Exemplos:

```
int a;
float c;
a = 5;
c = 67.89505456;
```

Comando de Atribuição

- O comando de atribuição pode conter expressões do lado direito:

variável = expressão ;

- Atribuir um valor de uma expressão a uma variável significa calcular o valor daquela expressão e copiar aquele valor para uma determinada variável.
- Exemplos:

```
int a;
float c;
a = 5+5+10;
c = 67.89505456+8-9;
```

Comando de Atribuição

No exemplo abaixo, a variável **soma** recebe o valor calculado da expressão **a + b**

```
soma = a + b;
```

Variável

Expressão

Atribuição

- O operador de atribuição é o sinal de igual (=)

À esquerda do operador de atribuição deve existir somente o nome de uma **variável**.

=

À direita, deve haver uma **expressão** cujo valor será calculado e armazenado na variável

Expressões Simples

Uma constante é uma expressão e como tal, pode ser atribuída a uma variável (ou em qualquer outro lugar onde uma expressão seja necessária).

- Ex1:

```
int a;
a = 10;
```

- Ex2:

```
char b;
b = 'F';
```

- Ex3:

```
double c;
c = '3.141592';
```

Expressões Simples

Uma variável também é uma expressão e pode ser atribuída a outra variável.

Ex:

```
int a, b;
a = 5;
b = a;
```

Expressões Simples

O endereço de uma variável também é uma expressão, e é obtido colocando-se o símbolo & antes do nome da variável.

Ex:

```
int *endereco;
int a;

a = 5;
endereco = &a;
```

Exemplos de atribuição

- **OBS:** Sempre antes de usar uma variável, esta deve ter sido **declarada**.

```
int a,b;
float f,g;
char h;

a = 10;
b = -15;
f = 10.0;
h = 'A';

a = b;
f = a;
a = (b+f+a);
```

Exemplos errados de atribuição

```
int a,b;
float f,g;
char h;

a b = 10;
b = -15
d = 90;
```

Estrutura Básica de um Programa em C

A estrutura básica é a seguinte:

```
Declaração de bibliotecas Usadas

Declaração de variáveis

int main(){
Declaração de variáveis

Comandos
.
.
.
Comandos
}
```

Estrutura Básica de um Programa em C

Exemplo:

```
#include <stdio.h>

int main(){
  int a;
  int b,c;

  a = 7+9;
  b = a+10;
  c = b-a;
}
```

Informações Extras: 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`
- 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 ($2e2 = 2 \cdot 10^2 = 200.0$)