

Computação 1

Variáveis

Prof. Luiz Fernando Carvalho

luizfcarvalho@utfpr.edu.br

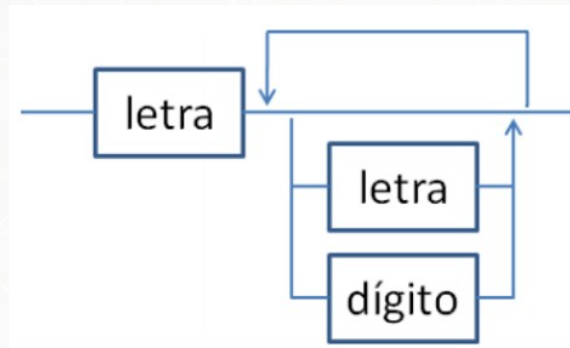
Variáveis

- Posição na memória do computador que é utilizada para armazenar temporariamente os dados que são utilizados pelo programa
- Devem ser declaradas antes da utilização

```
tipo nome_variavel;
```

- Declaração de variáveis pode ser feita:
 - Escopo global:
 - Variáveis declaradas fora das funções
 - Escopo local
 - Variáveis declaradas dentro das funções ou como parâmetros das funções

Variáveis



Nomes das variáveis:

1. Devem começar com uma **letra**;
2. Os próximos caracteres podem ser **letras** ou **números**;
3. Não pode utilizar nenhum símbolo, exceto **underline**;
4. Não pode conter **espaços em branco**;
5. Não pode conter letras com **acentos**;

Linguagem C – Palavras reservadas

- Não podem ser utilizadas como nome de variável

Lista de palavras-chave da linguagem C

auto	double	int	struct	break	else	long	switch
case	enum	if	typeof	continue	float	return	while
union	const	for	short	unsigned	char	extern	signed
void	default	do	sizeof	volatile	goto	register	static

BACKES, A. Linguagem C: completa e descomplicada

Armazenamento de dados

- É preciso estabelecer um tamanho padronizado que as variáveis devem possuir na memória do computador
 - Sem um tamanho padrão, como saberíamos onde começa e onde termina cada variável?

Armazenamento de dados

- Há cinco tipos **básicos** de dados em C
 - Caracteres (**char**)
 - Inteiro (**int**)
 - Ponto flutuante (**float**)
 - Precisão dupla (**double**)
 - Sem valor (**void**)
- O tamanho e a faixa desses tipos de dados variam de acordo com o tipo de processador e com a implementação do compilador
- Números inteiros **COSTUMAM** ser armazenados em espaços de 4 bytes, enquanto caracteres são armazenados em 1 byte.

Armazenamento de dados

- Geralmente...

Tipo	Bytes	Faixa de valores
char	1	-128 a 127 ou 0 a 255
int	4	-2.147.483.648 a 2.147.483.647
usingned int	4	0 a 4.294.967.295
short int	2	-32.768 a 32.767
unsigned short int	2	0 a 65.535
long int	4	Mesmo que int
unsigned long int	4	Mesmo que o unsigned int
float	4	1,17E-38 a 3,4E+38
double	8	2,2E-308 a 1,79E+308
long double	10	3,4E-4932 a 1,1E+4932

Tabela ASCII

Tipos de variáveis

```
double gasolina = 3.58741;  
char letra = 'a';  
short int idade = 22;  
float preco = 1.99;
```

- Considerações

- C é case-sensitive: diferencia maiúscula de minúscula
 - São diferentes: Preço, preco, pReco, prEco, preCo, precO, PRECO
- Por padrão `float` e `double` usam o . para representar decimal
 - Adotam padrão: 32-bits e 64-bits IEEE 754 de precisão

Números sem . são interpretados por padrão como `int`
Números com . são interpretados por padrão como `double`

Inteiros

unsigned

Tipo	Bytes	Maior valor
short	2	65.535
int, long	4	4.294.967.295
long long	8	18.446.744.073.709.551.615

Em C é possível armazenar valores tanto positivos e negativos (com sinal) quanto só positivos sem sinal. Usa-se a cláusula `signed` e `unsigned int` para designar com sinal e sem sinal, respectivamente.

signed

Tipo	Bytes	Maior valor
short	2	-32.768 a 32.767
int, long	4	-2.147.483.648 a 2.147.483.647
long long	8	-9.223.372.036.854.755.808 a 9.223.372.036.854.775.807

Todos esses subtipos podem (e costumam) ser abreviados, tirando deles a palavra `int`. Por exemplo, `short int` geralmente é escrito apenas como `short`.

Ponto flutuante

- Em C, os números reais (ponto flutuante) são definidos em dois tipos
 - `float` (4 bytes): precisão simples
 - `double` (8 bytes): precisão dupla
- Podem ser escritos em forma de notação científica

3.14159e-7
 3.14159×10^{-7}

1.234E+26
 1.234×10^{26}

4.56e5
 4.56×10^5

A letra **e** pode ser minúscula ou maiúscula

Ponto flutuante

- Geralmente...

Tipo	Intervalo
float (4 bytes)	10^{-38} a 10^{38}
double (8 bytes)	10^{-308} a 10^{308}
long double (12 bytes)	10^{-4932} a 10^{4932}

E se houver números além da capacidade de um `long double`, ou número complexos?

- Número complexos, exemplo: `long double _Complex` (ver manual C99 e C11);
- Biblioteca `openssl`, para números muito grandes, usados em criptografia por exemplo...

Tipos de variáveis

- O tamanho ocupado pelos tipos de variáveis
 - `sizeof(tipo ou nome_variável);`

```
int a = 1234567;  
printf("Um char ocupa %i byte\n", sizeof(char));  
printf("Um int ocupa %i bytes\n", sizeof(int));  
printf("A variavel \"a\" ocupa %i bytes\n", sizeof(a));  
printf("Um float ocupa %i bytes\n", sizeof(float));  
printf("Um double ocupa %i bytes\n", sizeof(double));  
printf("Um short int ocupa %i bytes\n", sizeof(short int));  
printf("Um long int ocupa %i bytes\n", sizeof(long int));  
printf("Um long double ocupa %i bytes\n", sizeof(long double));
```

```
Um char ocupa 1 byte  
Um int ocupa 4 bytes  
A variavel "a" ocupa 4 bytes  
Um float ocupa 4 bytes  
Um double ocupa 8 bytes  
Um short int ocupa 2 bytes  
Um long int ocupa 8 bytes  
Um long double ocupa 16 bytes
```

Constantes

- A linguagem C permite que programador especifique nomes a constantes usadas no programa
- As constantes melhoram a organização do código

Sintaxe:

```
#define NOME valor
```

Exemplo:

```
#define PI 3.1415
```

- Variável como constante

Sintaxe:

```
const tipo nome = valor;
```

Exemplo:

```
const double PI = 3.1415;
```

Constantes

```
#include<stdio.h>

#define LUZ 299792458
#define PI 3.14159265

int main(){
    double energia;
    float massa = 0.02, raio = 2.0, area;


    energia = massa * LUZ * LUZ; //e = m *c^2
    printf("Segundo Einstein, a energia do corpo e': %lf", energia);

    area = PI * raio * raio;
    printf("\n\n A area de um circulo com raio %f e': %f", raio, area);

    return 0;
}
```


Operadores Aritméticos

- Permite realizar “contas” com os valores das variáveis e constantes



Operador	Operação
()	Parênteses
%	Resto da divisão entre inteiros
*	Multiplicação
/	Divisão
+	Soma
-	Subtração

$$4 \% 2 = 0$$

$$5 \% 2 = 1$$

$$17 \% 3 = 2$$

$$100 \% 50 = 0$$

Conversão explícita de tipos

- As operações entre valores do tipo ponto flutuante ocorrem normalmente como as de números inteiros
- Em operações mistas, o inteiro é convertido (momentaneamente) para ponto flutuante

```
float x;  
x = 1/7;  
printf("%f", x);
```

0.000000

Divisão entre inteiros, gera um resultado inteiro e esse valor é guardado numa variável float (convertido para float)

```
float x;  
x = 1.0/7;  
x = 1/7.0;  
x = 1.0/7.0;  
printf("%f", x);
```

0.142857

Conversão explícita de tipos

- A conversão explícita de tipos de variáveis é chamado *casting*

(novo tipo) operação

```
int a = 5, b = 2;  
float x;
```

```
x = (float) a/b;  
printf("%f", x);
```

O valor da operação a/b é convertido para `float`

Conversão explícita de tipos

- Em uma operação, o resultado sempre será convertido implicitamente para o tipo de maior abrangência.

Sem conversão

```
int a=5, b=2;  
float x;  
  
x = a / b;  
printf("%f", x);
```

2.000000

Conversão implícita

```
int a=5;  
float x, b=2.0;  
  
x = a / b;  
printf("%f", x);
```

2.500000

Conversão explícita

```
int a=5, b=2;  
float x;  
  
x = (float) a / b;  
printf("%f", x);
```

2.500000

Saída de dados

- Códigos usados na função `printf()`

Código	Formato
<code>%c</code>	Caractere
<code>%d</code> ou <code>%i</code>	Inteiro decimal com sinal
<code>%e</code>	Notação científica
<code>%f</code>	Ponto flutuante decimal
<code>%g</code>	Menor representação entre <code>%f</code> e <code>%e</code>
<code>%o</code>	Octal sem sinal
<code>%s</code>	<i>String</i> de caracteres
<code>%u</code>	Inteiro sem sinal (<code>unsigned int</code>)
<code>%x</code>	Hexadecimal sem sinal
<code>%%</code>	Símbolo %

Caracteres de escape no printf

Caractere	Significado
\a	Caractere (invisível) de aviso sonoro (bip)
\n	Caractere (invisível) de nova linha
\t	Caractere (invisível) de tabulação horizontal
\\	Caractere de barra invertida \'
\'	Caractere de aspas simples
\''	Caractere de aspas duplas
\?	Sinal de interrogação

Obs.: Não deve conter espaço entre \ e o símbolo desejado

Saída de dados

- Formato de impressão dos números reais
 - Junto com o `%f` uma especificação de quantas casas decimais se deseja que o número tenha;
 - Especifica-se também o número total de caracteres do número a ser impresso;

```
printf("%6.3f", x);
```

- Define que se quer imprimir um `float` com **3 casas decimais** e com um **tamanho total** de 6 caracteres.

Saída de dados

- Formato de impressão dos números reais

```
printf("%.3f", x);
```

- O número de casas decimais é sempre respeitado;
- O tamanho total inclui o ponto decimal e um eventual sinal de menos (-);
- Se a soma do número de caracteres da parte inteira, mais o ponto decimal, mais a parte fracionária, mais um eventual sinal de menos ainda for menor do que o tamanho total especificado no formato, então, espaços em branco serão acrescentados à esquerda da parte real do número;
- Se o tamanho total de caracteres for maior do que o tamanho total especificado no formato, então apenas o número de casas decimais é respeitado.

Exercícios

- 1) Peça a altura (em metros) e o peso (Kg) de uma pessoa. Calcule e mostre o IMC da pessoa

$$imc = \frac{peso}{altura^2}$$

Dica: para usar potência, deve-se inserir a biblioteca `math.h` no começo do arquivo e usar a função `pow`.

- 2) Calcule a média aritmética de 4 números reais que o usuário digitar. Imprima a média na tela apenas com 2 casas decimais.