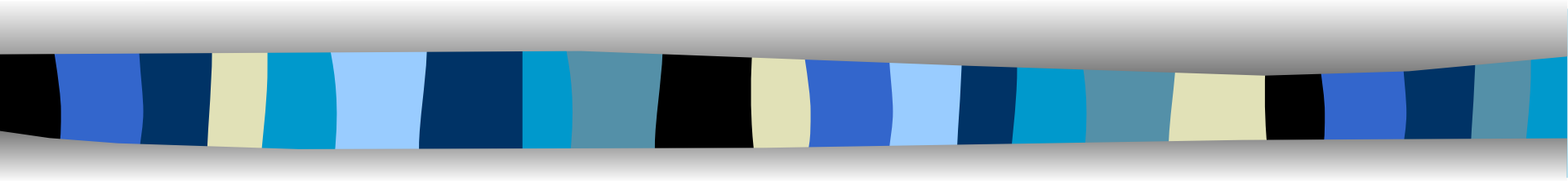
Variáveis, Constantes e Operadores



João Frederico Roldan Viana

jfredrv@gmail.com

(85)99231.2777

Agenda

- Variáveis, Constantes e Operadores
  - Tipos de Dados Básicos
  - Modificadores
  - Identificadores
  - Palavras Reservadas
  - Variáveis
  - Constantes
  - Operadores

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos
  - char
    - O valor armazenado é um caracter.
    - Caracteres são armazenados em um byte.
    - Caracteres geralmente são armazenados em códigos (usualmente o código ASCII - American Standard for Information Interchange).

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos
  - char (cont.)
    - Conjunto de caracteres ASCII:

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	‘
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	¡	=	¿	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	’	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	—	}	~	del		

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos
  - char (cont.)
    - Conjunto de códigos especiais ASCII:

Carac	Descrição	Carac	Descrição
nul	Caractere nulo	soh	Começo de cabeçalho de transmissão
stx	Começo de texto	etx	Fim de texto
eot	Fim de transmissão	enq	Interroga
ack	Confirmação	bel	Sinal sonoro
bs	Volta um caractere	ht	Tabulação horizontal
lf	Passa para próxima linha	vt	Tabulação vertical
ff	Passa para próxima página	cr	Passa para início da linha
so	Shift-out	si	Shift-in
dle	Data line escape	dc1	Controle de dispositivo
dc2	Controle de dispositivo	dc3	Controle de dispositivo
dc4	Controle de dispositivo	nak	Negativa de confirmação
syn	Synchronous idle	etb	Fim de transmissão de um bloco
can	Cancela	em	Fim de meio de transmissão
sub	Substitui	esc	Escape
fs	Separador de arquivo	gs	Separador de grupo
rs	Separador de registro	us	Separador de unidade
sp	Espaço em branco		

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos
  - int
    - O valor armazenado é um número inteiro.
    - O tamanho do subconjunto que pode ser representado pelo computador normalmente depende da máquina em que o programa está rodando.
    - Atualmente em C os números inteiros são armazenados em 32 bits.

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos
  - float
    - O valor armazenado é um número em ponto flutuante de precisão simples.
    - Normalmente são armazenados em 32 bits.
    - São conhecidos como números reais, no entanto, os computadores somente podem armazenar e trabalhar com uma parte limitada do conjunto dos números reais.

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos
  - double
    - O valor armazenado é um número em ponto flutuante de precisão dupla.
    - São armazenados em 64 bits.
  - void
    - Este tipo serve para indicar que um resultado não tem um tipo definido.
    - Uma das aplicações deste tipo em C é criar um tipo vazio que pode posteriormente ser modificado para um dos tipos anteriores.

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos

Tipo		Palavra-chave
Caractere		char
Inteiro		int
Ponto flutuante	precisão simples	float
	precisão dupla	double
Sem valor		void

Variáveis, Constantes e Operadores

- Tipos de Dados Básicos

Tipo	bits	Valor Mínimo	Valor Máximo
char	8	-2^7 (-128)	2^7-1 (127)
int	32	-2^{31} - 2.147.483.648	$2^{31}-1$ 2.147.483.647
float	32	Seis dígitos de precisão	
double	64	Dez dígitos de precisão	

Variáveis, Constantes e Operadores

- Modificadores
  - São palavras que alteram o tamanho do conjunto de valores que o tipo pode representar.
  - Exemplos do uso de modificadores
    - Permitir que possam ser usados mais do que 32 bits para armazenar números inteiros.
    - Obrigar uma variável armazenar apenas números inteiros sem sinal.

Variáveis, Constantes e Operadores

- Modificadores
  - Tipos de modificadores
    - **unsigned**: Aplicado aos tipos **int** e **char**. Faz com que o bit de sinal não seja usado, ou seja o tipo passa a ter um bit a mais.
    - **signed**: Aplicado aos tipos **int** e **char**. O uso de signed com **int** é redundante.
    - **long**: Aplicado aos tipos **int** e **double**. Aumenta o número de bytes para armazenamento.
    - **short**: Aplicado aos tipos **int** e **double**. Reduz o número de bytes para armazenamento.

Variáveis, Constantes e Operadores

- Modificadores para char

Tipo		bits	Valor Mínimo	Valor Máximo
	char	8	-128	127
signed				
unsigned			0	255

Variáveis, Constantes e Operadores

- Modificadores para int

Tipo		bits	Valor Mínimo	Valor Máximo
	int	32	- 2.147.483.648	2.147.483.647
signed				
unsigned			0	4.294.967.295
short		16	-2^{15} (-32.768)	$2^{15}-1$ (32.767)
signed short				
unsigned short			0	65.535
long		32	mesmo que int	
signed long				
unsigned long			mesmo que unsigned int	
long long		64	-2^{63} ($-9{,}2\times10^{18}$)	$2^{63}-1$ ($9{,}2\times10^{18}$)

Variáveis, Constantes e Operadores

- Modificadores para char

Tipo		bits	Valor Mínimo	Valor Máximo
	char	8	-128	127
signed				
unsigned			0	255

Variáveis, Constantes e Operadores

- Modificadores para double

Tipo		bits	Precisão
	double	64	Dez dígitos
long		80	

Variáveis, Constantes e Operadores

- Identificadores
  - São os nomes das variáveis e funções usadas no programa.
  - Devem seguir as seguintes regras de construção:
    - Começar por uma letra (a - z , A - Z) ou um underscore (_).
    - O resto do identificador deve conter apenas letras, underscores ou dígitos (0 - 9).
    - Em C, podem ter até 32 caracteres.
    - Respeitar maiúsculas e minúsculas.
    - Não podem ser iguais a uma palavra reservada.
    - Não devem ser iguais a funções da biblioteca C.

Variáveis, Constantes e Operadores

- Palavras Reservadas
  - São os nomes de uso restrito da linguagem (comandos, estruturas, declarações, etc.).
  - Não podem sem usadas com identificadores.

Variáveis, Constantes e Operadores

- Palavras Reservadas

asm	do	friend	pascal	public
auto	double	goto	private	register
break	-ds	huge	protected	return
case	else	if	template	_saveregs
cdecl	enum	inline	this	_seg
char	_es	int	typedef	short
class	extern	interrupt	union	signed
const	_export	_loadds	unsigned	sizeof
continue	far	long	virtual	_ss
_cs	_fastcall	near	void	static
default	float	new	volatile	struct
delete	for	operator	while	switch

Variáveis, Constantes e Operadores

- Variáveis
  - São nomes dados para posições de memória a fim de facilitar o manuseio dos dados durante a criação dos programas.
  - Regras para os nomes das variáveis
    - Começar por uma letra (a - z , A - Z) ou um underscore (_).
    - O resto do identificador deve conter apenas letras, underscores ou dígitos (0 - 9).
    - Respeitar maiúsculas e minúsculas.
    - Não pode usar palavras reservadas.

Variáveis, Constantes e Operadores

- Variáveis
  - Recomendações para os nomes das variáveis
    - Utilizar nomes que identifiquem o propósito da variável
    - Não utilizem variáveis com o nome todo em letras maiúsculas, uma vez que esse é uma prática usada para constantes.
    - Variáveis simples devem ser escritas todas com letras minúsculas.
    - Variáveis compostas devem iniciar com maiúscula a partir da segunda palavra.

Variáveis, Constantes e Operadores

- Variáveis
  - Declaração de Variáveis
    - Para usar uma variável em um programa, é necessário fazer sua declaração antes.
    - Informa ao processador quais são os nomes utilizados para armazenar dados variáveis e quais são os tipos usados.
    - Deste modo o processador pode alocar (reservar) o espaço necessário na memória para a manipulação destas variáveis.
    - É possível declarar mais de uma variável ao mesmo tempo, basta separá-las por vírgulas.

Variáveis, Constantes e Operadores

- Variáveis
  - Declaração de Variáveis (cont.)
    - A sintaxe para declaração de variáveis é:

      [modificador] tipo nome_var1[, nome_var2, . . .];
  - Exemplos de declarações de variáveis
    - int numInteiro;
    - int i, j, k;
    - char letra;
    - float nota1, nota2, media;
    - double numReal;
    - unsigned int numInteiroSemSinal;

Variáveis, Constantes e Operadores

- Variáveis
  - Escopo
    - Onde uma variável é definida.
    - Determina a região em que ela poderá ser usada.
    - Tipos:
      - Local
      - Parâmetro Formal
      - Global

Variáveis, Constantes e Operadores

- Variáveis
  - Escopo Local
    - Só podem ser utilizadas dentro do bloco que foram definidas.
    - Criada no início do bloco e "destruída" no final.
    - A memória só será alocada quando necessária.
    - Evita problemas em outras partes do código.

Exemplo

```
void atribui1()
{
  int x;
  x = 20;
}
```

```
void atribui2()
{
  int y;
  y = -50;
}
```

Variáveis, Constantes e Operadores

- Variáveis
  - Escopo de Parâmetro Formal
    - Definidas depois do nome da função e entre parênteses.
    - São os valores que serão passados para uma função.

Exemplo

```
void calculaSoma(float nota1, nota2)
{
    soma = nota1 + nota2;
}
```

Variáveis, Constantes e Operadores

- Variáveis
  - Escopo Global
    - São definidas nas declarações globais;
    - Reconhecidas no código inteiro;
    - Podem ser usadas em qualquer lugar do código, pois guardam os seus valores durante toda a execução do programa;
    - Alocadas quando o programa inicia e “destruídas” quando o programa termina.

Variáveis, Constantes e Operadores

- Variáveis
  - Modificadores

Tipo	Significado	Exemplo
const	Tratada como constante	**const int ref = 14;**
static	Não perde o valor ao sair do escopo	**static int a;**

Variáveis, Constantes e Operadores

- Variáveis
  - Inicialização

Exemplo1

```
char letra = 'a';
int primeiro = 0;
float num = 123.23;
```

Exemplo2

```
int valor;
valor = 25;
```

Variáveis, Constantes e Operadores

- Constantes
  - Valores que não podem sem alterados pelo programa.
  - Tipos básicos de constantes:
    - Inteira;
    - Ponto Flutuante;
    - Caractere;
    - Cadeia de Caracteres.

Variáveis, Constantes e Operadores

- Constantes
  - Constante Inteira:
    - Número de valor inteiro
    - Seqüência de dígitos decimais
    - Exemplos:
      - ✓ `const int num1 = 0;`
      - ✓ `const int num2 = -45;`
      - ✓ `const int num3 = 1010;`

Variáveis, Constantes e Operadores

- Constantes
  - Constante Ponto Flutuante:
    - Número de valor real
    - Parte inteira separada da fracionária por ponto (.)
    - Exemplos:
      - ✓ `const float num1 = 0.123;`
      - ✓ `const float num2 = .76;`
      - ✓ `const float num3 = 1.2e3;`
      - ✓ `const float num4 = 10.6E2;`
      - ✓ `const float num5 = -1e-1;`
      - ✓ `const float num6 = 345.;`
      - ✓ `const float num7 = 67;`

Variáveis, Constantes e Operadores

- Constantes
  - Constante Caractere:
    - Caractere delimitado por aspas simples (‘’)
    - Número da tabela ASCII
    - Caractere especial precedido da barra invertida
    - Exemplos:
      - ✓ const char caractere1 = ‘A’;
      - ✓ const char caractere2 = ‘1’;
      - ✓ const char caractere3 = 65;
      - ✓ const char caractere4 = ‘\n’;
      - ✓ const char caractere5 = ‘\”;
      - ✓ const char caractere6 = ‘\\’;

Variáveis, Constantes e Operadores

- Constantes
  - Constante Cadeia de Caracteres:
    - Seqüência de caracteres delimitados por aspas duplas (“”)
    - Caractere especial precedido da barra invertida
    - Exemplos:
      - ✓ const char *cadeia1 = “A”;
      - ✓ const char *cadeia2 = “1”;
      - ✓ const char *cadeia3 = “Linha 01”;
      - ✓ const char *cadeia4 = “Linha 01/nLinha 02”;

Variáveis, Constantes e Operadores

- Operadores
  - Responsáveis pelas operações elementares de uma linguagem.
  - Aplicados a um ou mais valores.
  - Podem ser de:
    - Atribuição
    - Aritméticos
    - Atribuição aritmética
    - Incrementais
    - Relacionais
    - Lógicos

Variáveis, Constantes e Operadores

- Operadores – Atribuição
  - Usado para transferir o resultado de uma expressão para uma variável.
  - Sintaxe
    - *identificador = expressao;*
  - Atribuições Múltiplas
    - *var_1 [= var_2] [= var_3 . . .] = expressao;*
  - Conversão de Tipo
    - Automática: int num1 = 5.7;
    - Explícita: float num2 = (int) (3.6 + 2.1);

Variáveis, Constantes e Operadores

- Operadores – Aritméticos
  - Sintaxe
    - *operando operador operando*
  - Operadores
    - + : adição
    - - : subtração
    - * : multiplicação
    - / : divisão
    - % : módulo (resto da divisão inteira)

Variáveis, Constantes e Operadores

- Operadores – Aritméticos
  - Divisão
    - O resto é trucado quando aplicado a números inteiros: 5 / 2 == 2
    - Caso se deseje o valor fracionário deve-se transformar pelo menos um número em ponto flutuante: 5 / 2.0 == 2.5 ou 5.0 / 2 == 2.5
    - Caso as variáveis sejam inteiras, usa-se o *casting*

Exemplo

```
float x;
int y = 5, z = 2;
x = y / (float) z;
```

Variáveis, Constantes e Operadores

- Operadores – Aritméticos
  - Resto
    - Não pode ser usado em ponto flutuante
  - Precedência
    - Multiplicação (*), divisão (/) e módulo (%) têm precedência sobre os operadores de adição.
    - Entre operadores de mesma precedência as operações são efetuadas da esquerda para a direita.

Variáveis, Constantes e Operadores

- Operadores – Atribuição aritmética
  - Usados para alterar uma variável realizando operação aritmética com ela.
  - Operadores
    - *var += exp;* (*var = var + exp;*)
    - *var -= exp;* (*var = var - exp;*)
    - *var *= exp;* (*var = var * exp;*)
    - *var /= exp;* (*var = var / exp;*)
    - *var %= exp;* (*var = var % exp;*)

Variáveis, Constantes e Operadores

- Operadores – Incrementais
  - Instruções chamadas de incremento e decremento.
  - Operadores
    - *++var (var = var + 1;)*
    - *var++ (var = var + 1;)*
    - *--var (var = var - 1;)*
    - *var-- (var = var - 1;)*

Variáveis, Constantes e Operadores

- Operadores – Incrementais
  - Se o operador for colocado á esquerda da variável, o valor da variável será incrementado (ou decrementado) antes que a variável seja usada em alguma outra operação.
  - Caso o operador seja colocado à direita da variável, o valor da variável será incrementado (ou decrementado) depois que a variável for usada em alguma outra operação.

Variáveis, Constantes e Operadores

- Operadores – Incrementais
  - No exemplo 1, y = ++x, significa que primeiro x terá seu valor incrementado depois y receberá o valor de x. Portanto, ao fim do programa, x será 11 e y será 11.
  - No exemplo 2, y = x++, significa que primeiro y receberá o valor de x, depois x terá seu valor incrementado. Portanto, ao fim do programa, y será 10 e x será 11.

Exemplo1

```
int x, y;
x = 10;
y = ++x;
```

Exemplo2

```
int x, y;
x = 10;
y = x++;
```

Variáveis, Constantes e Operadores

- Operadores – Relacionais
  - Verificam a relação de magnitude e igualdade entre dois valores.
  - Operadores
    - > : maior que
    - < : menor que
    - >= : maior ou igual a
    - <= : menor ou igual a
    - == : igual a
    - != : diferente de

Variáveis, Constantes e Operadores

- Operadores – Relacionais
  - Sintaxe
    - *expressao_1 operador expressao_2*
  - O resultado de uma expressão lógica é um valor numérico:
    - expressão verdadeira recebe o valor 1
    - expressão falsa recebe o valor 0.
  - Precedência
    - Os operadores relacionais de igualdade tem precedência menor que os de magnitude que tem precedência menor que os operadores aritméticos.
    - Operadores relacionais de mesma precedência são avaliados da esquerda para a direita.

Variáveis, Constantes e Operadores

- Operadores – Lógicos
  - Definem as maneiras como as relações podem ser conectadas.
  - Operadores
    - && (AND) : *expressao_1 && expressao_2*
    - || (OR) : *expressao_1* || *expressao_2*
    - ! (NOT) : ! *expressao_1*

Variáveis, Constantes e Operadores

- Operadores – Lógicos
  - O resultado de uma expressão lógica é um valor numérico:
    - expressão verdadeira recebe o valor 1
    - expressão falsa recebe o valor 0.
  - Precedência
    - O operador && tem precedência sobre o operador ||. Estes dois têm precedência menor que os operadores relacionais.
    - O operador ! tem a mesma precedência que os operadores incrementais.

Variáveis, Constantes e Operadores

- Operadores – Tabela de precedências

Categoria	Operadores
Parênteses	( )
Função	nome( )
Incremental, Lógico	++ , -- , !
Aritmético	* , / , %
Aritmético	+ , -
Relacional	< , > , <= , >=
Relacional	== , !=
Lógico	&&
Lógico	||
Atribuição	= , += , -= , *= , /= , %=