

Estrutura de um Programa C

Programa C

- Diretivas ao Pré-Processador

- Includes
- Macros

- Declarações Globais

- Funções
- Variáveis

- Definição das Funções

```
main ()  
{ /* begin */  
} /* end */
```

Variáveis

- variáveis em um programa C estão associadas a posições de memória que armazenam informações
- nomes de variáveis podem ter vários caracteres
- em C, apenas os 31 primeiros caracteres são considerados
- o primeiro caracter tem que ser uma letra ou *underscore* “_”
- o restante do nome pode conter letras, dígitos e sublinhados

Exercício

- Escolha a opção que inclui somente nomes válidos para variáveis na linguagem C.
 - a). If, a_b_2, H789, _yes
 - b). i, j, int, obs
 - c). 9xy, a36, x*y, --j
 - d). 2_ou_1, \fim, *h, j
 - e). Nenhuma das opções anteriores

Variáveis

- Exemplos de nomes de variáveis:

Corretos

Contador

Teste23

Alto_Paraiso

__sizeint

Incorretos

1contador

oi!gente

Alto..Paraíso

_size-int

Variáveis

- Palavras-chave de C não podem ser utilizadas como nome de variáveis: *int*, *for*, *while*, etc...
- C é *case-sensitive*:
contador \neq Contador \neq CONTADOR

Tipos de Dados

- O *tipo* de uma variável define os valores que ela pode assumir e as operações que podem ser realizadas com ela
- Ex:
 - variáveis tipo *int* recebem apenas valores inteiros
 - variáveis tipo *float* armazenam apenas valores reais

Tipos de dados básicos em C

- **char**: um byte que armazena o código de um caracter do conjunto de caracteres local
- **int**: um inteiro cujo tamanho depende do processador, tipicamente 16 ou 32 bits
- **float**: um número real com precisão simples
- **double**: um número real com precisão dupla

Modificadores de Tipos

- modificadores alteram algumas características dos tipos básicos para ajusta-los a necessidades específicas
- Modificadores:
 - **signed**: indica número com sinal (inteiros e caracteres)
 - **unsigned**: número apenas positivo (inteiros e caracteres)
 - **long**: aumenta abrangência (inteiros e reais)
 - **short**: reduz a abrangência (inteiros)

Abrangência dos Dados: 16 bits

Tipo	Tamanho(bytes)	Abrangência		
char	1	-128	a	127
unsigned char	1	0	a	255
int	2	-32768	a	32767
unsigned int	2	0	a	65535
short int	2	-32768	a	32767
long int	4	-2.147.483.648	a	2.147.483.647
unsigned long	4	0	a	4.294.967.295
float	4	$-3,4 \cdot 10^{38}$	a	$3,4 \cdot 10^{38}$
double	8	$-1,7 \cdot 10^{308}$	a	$1,7 \cdot 10^{-308}$
long double	10	$-3,4 \cdot 10^{4932}$	a	$3,4 \cdot 10^{4932}$

Constantes

- Constantes são valores fixos que não podem ser modificados pelo programa

Tipo		Exemplos
char	->	'a' '\n' '9'
int	->	123 1 1000 -23
long int	->	35000L -45L
short int	->	10 -12 90
unsigned int	->	1000U 234U 4365U
float	->	123.45F 3.1415e-10F
double	->	123.45 -0.91254

Constantes char

Barra invertida

- \a bip
- \b backspace
- \n newline
- \t tab horizontal
- \' apóstrofe
- \" aspa
- \\ backslash
- \f form feed

Declaração de Variáveis

- A declaração de uma variável segue o modelo:

TIPO_VARIÁVEL lista_de_variaveis;

- Ex:

```
int x, y, z;
```

```
float f;
```

```
unsigned int u;
```

```
long double df;
```

```
char c = 'A';          /* variavel definida e iniciada */
```

```
char s[ ] = "vetor de caracteres";
```

Atribuição de Variáveis

- Forma geral:

variavel = expressão ou constante

Múltiplas atribuições

- C permite a atribuição de mais de uma variável em um mesmo comando:

$x = y = z = 0;$

Conversões de Tipos na Atribuição

- Quando uma variável de um tipo é atribuída a uma de outro tipo, o compilador automaticamente converte o tipo da variável a direita de “=” para o tipo da variável a esquerda de “=”

- Ex:

```
int i; char ch; float f;
```

```
ch = i;    /* ch recebe 8 bits menos significativos de x */
```

```
i = f;    /* x recebe parte inteira de f */
```

```
f = ch;    /* f recebe valor 8 bits convertido para real */
```

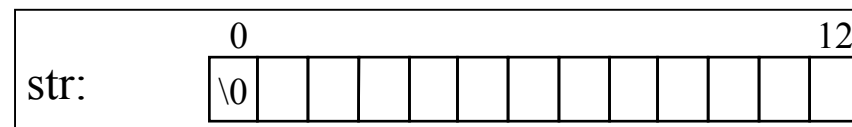
```
f = i;    /* idem para inteiro i */
```

Strings

- strings são seqüências de caracteres adjacentes na memória. O caractere ‘\0’ (= valor inteiro 0) indica o fim da seqüência

Ex: `char str[13];`

- define um string de nome “str” e reserva para ele um espaço de 13 (12 + ‘\0’) bytes na memória

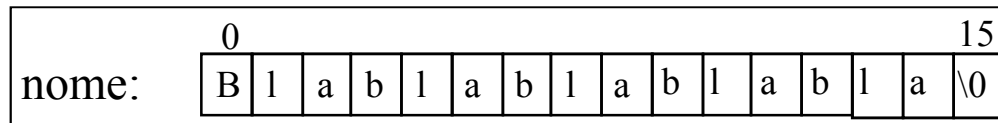


Strings

Ex:

```
char nome[16] = "Blablablabla";
```

- define um string de nome “nome”, reserva para ele um espaço de memória de 16 (15 + ‘\0’) bytes e inicia-o com o texto indicado



Strings

- os caracteres podem ser individualmente acessados por indexação:
- Ex:

```
nome[0] = 'B';  
nome[10] = 'l'
```

	0														15	
nome:	B	l	a	b	l	a	b	l	a	b	l	a	b	l	a	\0

Exercícios

- O trecho de programa a seguir é

```
main( )  
{  
    char condicao;  
    condicao = 'D';  
    int i = 1;  
}
```

- a). Válido na linguagem C
 - b). Não válido na linguagem C
- Em C, "t" e 't' representam a mesma constante.
 - a). Verdadeiro
 - b). Falso

Escopo de Variáveis

- Escopo define **onde** e **quando** uma variável pode ser usada em um programa
- variável declarada **fora** das funções tem escopo de arquivo:

```
#include <stdio.h>  
int i;                /* variavel global */  
                    /* visivel em todo arquivo */  
void incr_i() { i++;}  
...  
void main() { printf("%d", i);}
```

Escopo de Variáveis

- Escopo de bloco: é visível apenas no interior do bloco

...

```
if (teste == TRUE) {
```

```
    int i;
```

```
    i = i+1;
```

```
    ...
```

```
}
```

```
else { i = i - 1; /* erro: i não definida */
```

```
    ...
```

```
}
```

...

Escopo de Variáveis

- Escopo de função: variável declarada na lista de parâmetros da função ou definida dentro da função
- Ex:...

```
int minha_fun (int x, int y) {  
  int i, j;  
  /* x,y,i e j visíveis apenas dentro da função */  
  ...  
}  
x = i+j; /* erro: x,i e j não definidos */
```

Expressões

- expressões são compostas por:
 - operandos: a, b, x, Meu_dado, 2, ...
 - operadores: +, -, %, ...
 - pontuação: (), { }, “,”

- Ex:

x

14

x + y

(x + y)*z + w - v

Expressões

- expressões retornam um valor:

`x = 5 + 4 /* retorna 9 */`

- esta expressão atribui 9 a x e retorna 9 como resultado da expressão

`((x = 5 + 4) == 9) /* retorna true */`

- na expressão acima, além de atribuir 9 a x, o valor retornado é utilizado em uma comparação

Expressões

- a ordem em que uma expressão é avaliada depende da prioridade dos operadores e da pontuação
- expressões podem aparecer em diversos pontos de um programa
 - comandos `/* x = y; */`
 - parâmetros de funções `/* sqrt(x + y); */`
 - condições de teste `/* if (x == y) */`

Operadores

Unários:

+ : mais unário ou positivo

```
/* + x; */
```

- : menos unário ou negação

```
/* - x; */
```

! : NOT ou negação lógica

```
/* ! x; */
```

& : endereço

```
/* &x; */
```

* : conteúdo (ponteiros)

```
/* (*x); */
```

++ : pré ou pós incremento

```
/* ++x ou x++ */
```

-- : pré ou pós decremento

```
/* -- x ou x -- */
```

Operadores

Binários:

+: adição de dois números	<code>/* x + y */</code>
- : subtração de dois números	<code>/* x - y */</code>
* : multiplicação de dois números	<code>/* x * y */</code>
/ : quociente de dois números	<code>/* x / y */</code>
?: resto da divisão:	<code>/* x % y */</code>

Atribuição

= : atribui

$x = y;$

+= : soma e atribui

$x += y; \Leftrightarrow x = x + y;$

-= : subtrai e atribui

$x -= y; \Leftrightarrow x = x - y;$

*= : multiplica e atribui

$x *= y; \Leftrightarrow x = x * y;$

/= : divide e atribui quociente

$x /= y; \Leftrightarrow x = x / y;$

%= : divide e atribui resto

$x \% = y; \Leftrightarrow x = x \% y;$

&= : E bit-a-bit e atribui

$x \& = y; \Leftrightarrow x = x \& y;$

|= : OU bit-a-bit e atribui

$x |= y; \Leftrightarrow x = x | y;$

<<= : shift left e atribui

$x \ll = y; \Leftrightarrow x = x \ll y;$

...

Atribuição

- Exemplos:

`x = 10;`

`y = 5;`

`x += y; /* x = 15 */`

`x -= 10; /* x = 5 */`

`x *= y; /* x = 35 */`

Exercícios

1- Diga o resultado das variáveis x, y e z depois da seguinte seqüência de operações:

```
int x, y, z;  
x=y=10;  
z=++x;  
x=-x;  
y++;  
x=x+y-(z--);
```

- a). $x = 11, y = 11, z = 11$
- b). $x = -11, y = 11, z = 10$
- c). $x = -10, y = 11, z = 10$
- d). $x = -10, y = 10, z = 10$
- e). Nenhuma das opções anteriores

2- Diga o resultado das variáveis x, y e z depois da seguinte sequência de operações:

```
int x,y;  
int a = 14, b = 3;  
float z;  
x = a/b;  
y = a%b;  
z = y/x;
```

- a). $x = 4.66666$, $y = 2$, $z = 0.4286$
- b). $x = 5$, $y = 2$, $z = 0.4$
- c). $x = 5$, $y = 2$, $z = 0$.
- d). $x = 4$, $y = 2$, $z = 0.5$
- e). $x = 4$, $y = 2$, $z = 0$.
- f). Nenhuma das opções anteriores

Operadores Relacionais

- Aplicados a variáveis que obedecem a uma relação de ordem, retornam 1 (true) ou 0 (false)

Operador

Relação

>

Maior do que

>=

Maior ou igual a

<

Menor do que

<=

Menor ou igual a

==

Igual a

!=

Diferente de

Precedência dos Operadores

Maior precedência

() [] ->

! ~ ++ -- . -(unário) (cast) *(unário) &(unário) sizeof

* / %

+ -

<< >>

<<= >>=

== !=

&

^

|

&&

||

?

= += -= *= /=

Menor precedência

Operadores Lógicos

- Operam com valores lógicos e retornam um valor lógico verdadeiro (1) ou falso (0)

Operador	Função	Exemplo
&&	AND (E)	(c >='0' && c <='9')
	OR (OU)	(a=='F' b!=32)
!	NOT (NÃO)	(!var)

Tabela Verdade

a	b	!a	!b	a && b	a b
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Exemplo: No trecho de programa abaixo o **if** será executado, pois o resultado da expressão lógica é verdadeiro:

```
int i = 5, j = 7;
```

```
if ( (i > 3) && (j <= 7) && (i != j) ) j++;
```

```
V AND V AND V = V
```

Exercícios

1- A operação lógica

$(-5 \parallel 0) \&\&(3 \geq 2) \&\&(1 \neq 0) \parallel (3 < 0)$ é:

- a). Verdadeira
- b). Falsa
- c). Inválida, pois sua sintaxe está errada.
- d). Nem Verdadeira nem Falsa
- e). Nenhuma das opções anteriores