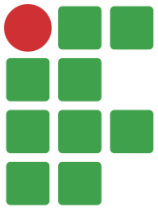
Introdução à Programação em C

Professor: Paulo de Tarso F. Júnior

paulodt@gmail.com



INSTITUTO FEDERAL
Paraíba

Roteiro

- Tópicos
  - 2.1 Introdução
  - 2.2 Um Programa C Simples: Impressão de uma Linha de Texto
  - 2.3 Outro Programa C Simples : Soma de Dois Inteiros
  - 2.4 Conceitos sobre Memória
  - 2.5 Aritmética em C
  - 2.6 Tomada de Decisões: Operadores de Igualdade e Relacionais

Introdução à Programação

- Livro-Texto
  - DEITEL, H. M. & DEITEL, P. J., Como Programar em C. LTC Editora, 1999.
  - Disponível em: www.paulodt.vai.la
- Bibliografia Complementar
  - ECKEL, BRUCE , Thinking in C++ . MindView Inc., 2000.
  - GACLI-UNICAMP, Introdução à Linguagem C.
  - JAMSA, KRIS & KLANDER, L., Programando em C/C++. Makron Books Editora do Brasil Ltda, 1999.

Introdução à Programação

- Linguagem de programação C
  - Abordagem estruturada e disciplinada para projeto de programa
- Programação estruturada
  - Capítulos 3 e 4 do livro-texto
  - Usada em todo o restante do livro-texto

Linguagens de Programação

```
1. /*Primeiro programa em C */
2. #include <stdio.h> /* biblioteca que realiza o “printf()” */
3. int main() {
4.     printf( “Bem-vindo ao C!\n" );
5.     return 0; /* como “main” foi declarada do tipo int tem que
   retornar um valor */
6.  }
```

- Comentários
  - Texto delimitado por **/*** e ***/** é ignorado pelo computador
  - Usado para descrever programa

Linguagens de Programação

- #include <stdio.h>
  - Diretiva do pré-processador
    - Indicação ao computador para carregar um certo arquivo
  - <stdio.h>
    - Permite operações padrão de entrada/saída
- int main()
  - Programas em C contém uma ou mais funções, uma das quais tem que ser exatamente ***main***
  - ***Parênteses*** são usados para indicar uma função

Linguagens de Programação

- *int main() {*
- ....
- *}*
  - **int** significa que main "retorna" um valor inteiro
  - Chaves (**{ }**) indicam a existência de um bloco
    - O corpo de todas as funções tem que está contido entre chaves

Linguagens de Programação

- **printf(“Bem-vindo ao C!\n");**
  - Instrui o computador a realizar uma ação
    - Especificamente, imprime a cadeia de caracteres entre aspas (“ ”)
  - A linha inteira equivale a um ***comando***
    - Todos os comandos têm que terminar com um ponto-e-vírgula (;)
  - Caractere de escape (\)
    - Indica que o ***printf*** deve fazer algo fora do comum
    - **\n** é o caractere nova-linha

Linguagens de Programação

- **return 0;**
  - Maneira de sair de uma função
  - **Return 0 (zero)** indica o programa terminou normalmente
  - Fecha-chave (**}**)
    - Indica que o fim do ***main*** foi encontrado

Linguagens de Programação

- *Linker*
  - Quando uma função é chamada, o *linker* a localiza na biblioteca
  - A insere no programa-objeto (.obj)
  - Se o nome da função for escrito incorretamente, o *linker* produzirá um erro, pois não será capaz de encontrar a função na biblioteca

Linguagens de Programação

```
1.  /*Programa de soma */
2.  #include <stdio.h>
3.  int main() {
4.      int int1, int2, soma; /* declaração */
5.      printf( “Entre com o primeiro inteiro:\n" ); /* prompt */
6.      scanf( "%d", &int1 ); /* lê um inteiro */
7.      printf( “Entre com o segundo inteiro:\n" ); /* prompt */
8.      scanf( "%d", &int2 ); /* lê um inteiro */
9.      soma = int1 + int2; /* atribui à soma */
10.     printf( “A soma eh igual a %d\n", soma ); /* imprime soma */
11.     return 0; /* indica que o programa foi bem-sucedido */
12. }
```

1. Entre com o primeiro inteiro:
2. 45
3. Entre com o segundo inteiro:
4. 72
5. Soma eh igual a 117

Linguagens de Programação

- Vide análise do programa anterior
  - Comentarios,#include <stdio.h> emain
- int int1, int2, soma;
  - Declaração de variáveis
    - **Variáveis** → posições na memória nas quais um valor pode ser armazenado
  - **int** significa que as variáveis podem guardar inteiros (*-1, 3, 0, 47)*

Linguagens de Programação

- Nomes de variáveis (identificadores)
  - int1, int2,soma
  - Identificadores consiste de letras, dígitos (**não** podem começar por dígitos) e sublinha(_)
    - *Case sensitive (maiúscula diferente de minúscula)*
- Declarações aparecem antes dos comandos executáveis
  - Se um comando executável referencia uma variável não declarada será produzido um erro de sintaxe (compilador)

Linguagens de Programação

- scanf("%d", &int1);
  - Obtém um valor do usuário
    - **scanf** usa a entrada padrão (comumente o teclado)
- Este comando **scanf** tem dois argumentos
  - **%d** → indica que o dado deve ser um inteiro decimal
  - **&int1** → posição na memória na qual a variável está armazenada
  - **&** → pode parecer confuso neste ponto
    - Por enquanto, lembrar apenas de adicioná-lo ao nome da variável sempre que usar o comando scanf

Linguagens de Programação

- scanf("%d", &inteiro1);
  - Quando o programa está sendo executado, o usuário responde ao comando **scanf** digitando um número e, em seguida, pressionando a tecla **enter (*return)***

Linguagens de Programação

- Operador de atribuição =
  - Atribui um valor para a variável
  - É um operador binário (tem dois operandos)
    - soma = variavel1 + variavel2;
    - soma **recebe** variavel1 + variavel2
  - Variável que recebe valor posicionada à esquerda
- printf("Soma eh igual a %d\n", soma);
  - Similar ao **scanf**
    - **%d** significa que um decimal inteiro será impresso
    - **soma** especifica qual inteiro será impresso
  - Cálculos podem ser realizados dentro de um comando **printf**
    - printf("Soma e %d\n", int1 + int2);

Linguagens de Programação

- Variáveis
  - Nomes de variáveis correspondem a posições (locações) a serem reservadas na memória
  - Toda variável tem um nome, um tipo, um tamanho e um valor
  - Toda vez que um valor é atribuído a uma variável (através de scanf, por exemplo), o valor anterior é substituído (e destruído)
  - A leitura de variáveis da memória não as altera

Linguagens de Programação

- Cálculos Aritméticos
  - Usa-se * para a multiplicação e / para a divisão
  - Divisão inteira → Truncamento do quociente
    - 7 / 5 tem como retorno 1
  - Operador de módulo (%) Retorno do resto
    - 7 % 5 tem como retorno 2

Linguagens de Programação

- Precedência de Operadores
  - Alguns operadores têm prioridade sobre outros durante a avaliação da expressão (e.g., multiplicação antes da adição)
    - Usa-se parênteses quando necessário
  - Exemplo
    - Encontre a média de três variáveis a,b e c
      - Não se usa a + b + c / 3
      - Usa-se (a + b + c) / 3

Linguagens de Programação

- Operadores Aritméticos

Operação em C	Operador Aritmético	Expressão algébrica	Expressão em C
Adição	+	f + 7	f + 7
Subtração	-	p – c	p - c
Multiplicação	*	mn	m * n
Divisão	/	Xx/y	x / y
Módulo	%	r mod s	r % s

Linguagens de Programação

- Regras de Precedência de Operadores

Operador(es)	Operação(ões)	Ordem de Avaliação (precedência)
( )	Parênteses	Avaliados primeiro. Se estiverem alinhados, a expressão no par mais interno é avaliada. Se houver vários a avaliação ocorre da esquerda para direita
* / mod	Multiplicação, Divisão ou Resto	Avaliados em segundo lugar. Se houver vários operadores no mesmo nível, a avaliação ocorre da esquerda para a direita
+ -	Adição ou Subtração	Avaliados em terceiro lugar. Se houver vários operadores no mesmo nível, a avaliação ocorre da esquerda para a direita.

Linguagens de Programação

- Comandos executáveis
  - Realização de ações (cálculos, entrada/saída de dados)
  - Tomada de decisões
    - Decisão de impressão "*passa" ou "falha" a partir de um teste de* condição
- Estrutura de controle ***if (condicao) {bloco}***
  - Versão simples neste ponto, maior detalhamento posteriormente
  - Se a condição for **verdadeira**, então o corpo do comando **if** será executado
    - Igual a **0** é falso, diferente de zero é verdadeiro
  - O controle sempre prossegue após o comando if

TOMADA DE DECISÃO: OPERADORES DE IGUALDADE E RELACIONAIS

Operador de Igualdade/Relacional Algébrico Padrão	Operador de Igualdade/ Relacional em C	Exemplo de Condição em C	Significado em C
Operadores de Igualdade			
=	==	X == y	X é igual a y
≠	!=	X != y	X é diferente de y
Operadores Relacionais			
<	<	X < y	X é menor que y
>	>	X > y	X é maior que y
≤	<=	X <= y	X é menor ou igual a y
≥	>=	X >= y	X é maior ou igual a y

Linguagens de Programação

```
1.  /* Usando comandos if, operadores
2.  relacionais, e operadores de igualdade */
3.  #include <stdio.h>
4.      int main() {
5.      int num1, num2;
6.      printf( "Entre com dois inteiros e eu lhe direi\n" );
7.      printf( “as relações que eles satisfazem: " );
8.      scanf( "%d%d", &num1, &num2 ); /* le dois inteiros */
9.      if ( num1 == num2 )
10.         printf( "%d é igual a %d\n", num1, num2 );
11.     if ( num1 != num2 )
12.         printf( "%d é diferente de %d\n", num1, num2 );
13.     if ( num1 < num2 )
14.         printf( "%d é menor que %d\n", num1, num2 );
15.     if ( num1 > num2 )
16.         printf( "%d é maior que %d\n", num1, num2 );
17.     if ( num1 <= num2 )
18.         printf( "%d é menor ou igual a %d\n", num1, num2 );
19.     if ( num1 >= num2 )
20.         printf( "%d é maior ou igual a %d\n", num1, num2 );
21.     return 0; /* indica que o programa foi bem-sucedido */
22. }
```

```
1.  Entre com dois inteiros, e eu lhe direi
2.  as relações que eles satisfazem : 22 12
3.  22 é diferente de 12
4.  22 é maior que 12
5.  22 é maior ou igual a 12
```

Linguagens de Programação

- Palavras-chave
  - Palavras reservadas especiais para C
  - Não podem ser usadas como identificadores ou nomes de variáveis

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

Declaração de Variáveis em C

- ***<tipo> <lista de variáveis>;***
  - 5 tipos básicos**: char, int, float, void, double**
  - modificadores de tipo do C**: signed, unsigned, long e short**
    - ***int inteiro1, n1, x;***
    - ***unsigned int i;***
    - ***char letra, controle, nome[30];***
    - ***float x;***

Linguagens de Programação

- *Tipo Bits Valor*
- _____
- Char 8 -128 a 127
- unsigned char 8 0 a 255
- signed char 8 -128 a 127
- Int 16 -32.768 a 32.767
- DECLARAÇÃO 27 DE VARIÁVEIS EM C
- unsigned int 16 0 a 65.535
- long int 32 -2.147.483.648 a 2.147.483.647
- unsigned long int 32 0 a 4.294.967.295
- Float 32 3.4E-38 a 3.4E+38
- Doublé 64 1.7E-308 a 1.7E+308
- long double 90 3.4E-4932 a 1.1 E+4932
- Void 0 sem valor

Linguagens de Programação

- *scanf (string_de_controle, lista_de_argumentos);*
- *scanf("%d", &inteiro1);*
- *scanf("%d %c", &n1,&x); /* le um inteiro e um caractere */*
- *scanf("%30s", &nome);*

Entrada em C

Código	Significado
%c	Ler um caractere
%d	Ler um inteiro
%f	Ler um número em ponto flutuante
%x	Ler um número em hexadecimal
%o	Ler um número em octal
%s	Ler uma cadeia de caracteres (string)

Saída em C

- *printf (string_de_controle,lista_de_argumentos);*
  - printf ("Teste %% %%") -> "Teste % %“
  - printf ("%f",40.348) -> "40.348“
  - printf ("%.2f",40.348) -> "40.35“
  - printf ("Um caractere %c e um inteiro %d",'D',120)
    - -> "Um caractere D e um inteiro 120“
  - printf (“Meu nome é %s",nome) -> “Meu nome é Paulo de Tarso“
    - printf ("%s%d%%","Juros de ",10) -> "Juros de 10%"

Saída em C

Código	Formatação
%c	Caractere
%d	Decimal
%f	Ponto flutuante
%u	Decimal sem sinal
%x	Hexadecimal
%o	Octal
%p	Apontador
%s	Cadeia de caracteres (string)
%e	Notação científica (exponencial)

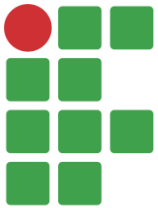
Saídas Especiais

Sequencia	Valor	Significado
‘\0’	0	Finalizador de um string (NULL)
‘\8’	0x07	Apito sonoro (*bell*)
‘\b’	0x08	Retrocesso (*backspace*)
‘\f’	0x0C	Avanço para o início de outra folha
‘\n’	0x0A	Avanço de linha (*new line*)
‘\r’	0x0D	Retorno de Cursor (*return*)
‘\\’	0x5C	Barra Invertida
‘\’’	0x27	Apóstrofo

Introdução à Programação em C

Professor: Paulo de Tarso F. Júnior

paulodt@gmail.com



INSTITUTO FEDERAL
Paraíba