

Introdução à Linguagem de Programação C

Simone de Oliveira Santos

10 de março de 2020



Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis
- 4 Operadores Aritméticos
- 5 Operadores Relacionais e Lógicos
- 6 Comandos de Entrada e Saída

Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis
- 4 Operadores Aritméticos
- 5 Operadores Relacionais e Lógicos
- 6 Comandos de Entrada e Saída

Introdução

- C foi criada por Dennnis Ritchie na década de 70
- É chamada de C porque suas principais características foram baseadas em outra linguagem chamada B



Figura: Dennis Ritchie em 2011.

Características

Principais características de C

- Linguagem de alto nível
- Poderoso conjunto de operadores e tipos de dados disponíveis
- Construções de controle de fluxo fundamentais

Características

Principais características de C

- Acesso a endereço de variáveis e a capacidade de fazer aritmética de endereços.
- Não provê operações para manipular diretamente objetos compostos, como cadeia de caracteres.
- Várias operações dependem de chamadas de funções externas.

Características

Principais características de C

- Paradigma de programação procedural
- Falta de proteção ao programador, por não possuir recursos de segurança
- Difícil detecção de alguns erros
- Portabilidade possibilitando que um código-fonte com poucas ou nenhuma modificação crie executáveis para diversas plataformas

Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis
- 4 Operadores Aritméticos
- 5 Operadores Relacionais e Lógicos
- 6 Comandos de Entrada e Saída

Execução de um programa em C

- Para ser executado, um programa C ele deve ser traduzido para linguagem de máquina correspondente ao modelo de computador usado.
- Esse processo é chamado de **compilação**
- C é uma linguagem compilada.

Compilação

Um programa compilador lê o programa-fonte escrito em C, e traduz cada uma de suas instruções gerando um programa-objeto.

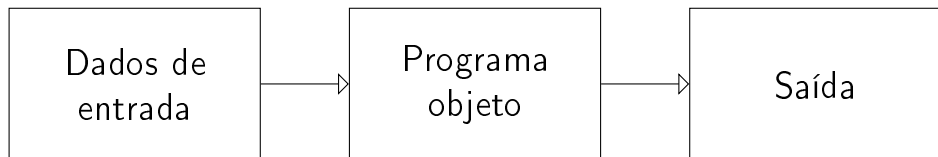


Execução de um programa em C

- A máquina em que o programa-objeto é executado não precisa ter um compilador instalado nem precisa ter acesso ao código C do programa.
- A construção de um programa que usa a linguagem C envolve duas fases independentes:
 - compilação
 - execução

Execução

A execução de um programa consiste em fornecer os dados de entrada ao programa-objeto e obter uma saída.



Primeiro programa em C

Pseudocódigo

```
1  algoritmo
2
3  escreva "Hello world!"
4
5  fim_algoritmo
```

Código-fonte em C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
```

Executando um programa em C

Comando para compilar

```
gcc nome_arquivo.c -o nome_arquivo
```

- após o comando gcc deverá ser colocado o nome do arquivo que deseja compilar.
- após o -o deverá ter o nome do arquivo que será gerado.

Comando para executar

```
./nome_arquivo
```

- chamar o nome do arquivo que foi gerado.

Sintaxe e Semântica de C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
```

- `include` é uma diretiva de pré-processador
- Linhas iniciadas por `#` são processadas antes do programa ser compilado
- `#include` inclui um arquivo de cabeçalho
- `#include <stdio.h>` é a biblioteca padrão do C

Sintaxe e Semântica de C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
```

- Cada instrução em C deve ser encerrada com ';'
- Deve-se preferencialmente escrever uma instrução por linha
- Um bloco de instruções são agrupadas por chaves { }

Sintaxe e Semântica de C

- Um programa em C tem de, obrigatoriamente, conter a função `main`.
- Ela é automaticamente chamada quando o programa é executado.
- As funções auxiliares são chamadas a partir da função principal.

Comentários de código

- Texto que pode ser inserido no código.
- O comentário não fará parte do programa, portanto não é lido pelo compilador.
- Sua principal função é permitir anotações que não vão interferir no código.
- Pode-se inserir comentários no código-fonte de duas formas.

Comentários de código

Comentário de bloco

Texto iniciado por `/*` e encerrado por `*/`

Comentário de linha

Texto com `//` no ponto inicial e encerra quando a linha é encerrada.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      /*isso é um comentário
6       ele não será lido pelo compilador*/
7      printf("Hello World!\n");
8      //isso também é um comentário
9      return();
10 }
```

Programa em C

Para desenvolver programas em C, é necessário um **editor de textos** e um **compilador**.

Editor de texto Local onde escreve-se os programas-fonte, salvos em arquivos geralmente com extensão .c.

Compilador Transforma os programas-fonte em programas-objeto (linguagem de máquina), a extensão dos executáveis varia de acordo com o SO.

Editores e compiladores

Editores de texto

- Code-blocks*
- Eclipse
- Dev C++ *
- NetBeans
- Visual Studio C++ *(pago)
- Notepad++
- Sublime

* Pode acompanhar o compilador

Compiladores

GCC Compilador para Linux
MinGW Versão do GCC para
Windows

Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis**
- 4 Operadores Aritméticos
- 5 Operadores Relacionais e Lógicos
- 6 Comandos de Entrada e Saída

Variáveis

- Todas as variáveis devem ser explicitamente declaradas
- Podem ser declaradas em qualquer parte do código.
- Sempre iniciado com letra **minúscula**.
- Nome de variável com mais de uma palavra recomenda-se que seja escrito como:

salarioDiretor
ou
salario_diretor

Variáveis

Declaração de variáveis em C segue o formato:

`tipo_da_variavel nome_da_variavel;`

- Mais de uma variável de um mesmo tipo, podem ser declaradas em uma linha. As variáveis são separadas por vírgula

```
4  int main()  
5  {  
6      int inteiro1, inteiro2, soma;  
7  
8      return 0;  
9  }
```


Variáveis

Regras para nomes de variáveis em C

- Formados apenas por letras, números e sublinhado _
- Pode iniciar com letra ou _
- Não pode conter palavras reservadas

Variáveis

Palavras reservadas de C

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

Variáveis

Regras para nomes de variáveis em C

Linguagem sensível ao caixa (*case sensitive*), ou seja, existe diferenciação entre letras maiúsculas e minúsculas.

```
4  int main()  
5  {  
6      int inteiro, Inteiro, inTEiro;  
7  
8      return 0;  
9  }
```

Tipos de Variáveis

Tipo	Tamanho	Faixa de Valores
char	1 byte	-128 a 127
unsigned char	1 byte	0 a 255
short	2 bytes	-32768 a 32767
unsigned short int	2 bytes	0 a 65535
int	4 bytes	-2147483648 a 2147483647
long int	4 bytes	-2147483648 a 2147483647
unsigned long int	4 bytes	0 a 4294967295

Tipo	Tamanho	Faixa de Valores	Precisão
float	4 byte	$1.2E^{-38}$ a $3.4E^{38}$	6 casas
double	8 bytes	$2.3E^{-308}$ a $1.7E^{308}$	15 casas
long double	10 bytes	$3.4E^{-4932}$ a $1.1E^{4932}$	19 casas

Variáveis

Tipos de variáveis em C

- Os caracteres são representados no computador como inteiros
- Eles são delimitados por aspas simples (' ')
- Cada caractere equivale a um número inteiro
(Ex: 'a' é o valor 97)
- Podemos atribuir um caractere para uma variável do tipo **int**

Variáveis

Tipos de variáveis em C

- Em C não existe o tipo lógico
- Caso deseja-se uma variável do tipo lógico utiliza-se uma variável do tipo inteiro, sendo

FALSO = 0

VERDADEIRO = 1 (ou qualquer valor diferente de zero)

Variáveis

Atribuição de valores

- Operador de atribuição: `=`
- Pode-se atribuir um mesmo valor para muitas variáveis em uma só instrução.

```
4  int main()  
5  {  
6      int num, valor, pontos;  
7  
8      num = 3;  
9      valor = pontos = 15;  
10  
11     return 0;  
12 }
```

Variáveis

Inicialização

É importante sempre inicializar as variáveis para evitar que elas assumam valores incorretos

```
4  int main()  
5  {  
6      int num, valor, pontos;  
7  
8      num = 3;  
9      valor = pontos = 15;  
10  
11     return 0;  
12 }
```

```
4  int main()  
5  {  
6      int num = 5;  
7  
8      return 0;  
9  }
```


Variáveis

Coersão

- Pode-se forçar a mudança do tipo de uma variável
- Usa operadores unários de coerção (conversão)

Variáveis

Seja o código a seguir:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      int n = 5;
6      float resultado;
7      resultado = n/2;
8
9      printf("Resultado: %f\n", resultado);
10     system("pause");
11     return 0;
12 }
13
```

Variáveis

O resultado obtido não é o esperado, para que isso seja possível deve ser feita a coersão como mostrado a seguir:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int n = 5;
6      float resultado;
7      resultado = (float) n/2;
8
9      printf("Resultado: %f\n", resultado);
10     system("pause");
11     return 0;
12 }
13
```

Constantes

- Semelhante a uma variável, mas o seu valor armazenado não muda durante a execução do programa
- Usado para dados que não devem ser mudados, evitando assim alterações acidentais dos dados
- São declaradas com a diretiva **#define** do pré-processador

Constantes

- O nome da constante deve ser em letras maiúsculas para um maior destaque no texto
- Durante a compilação, as ocorrências do nome da constante são substituídas pelo seu valor
- Não utiliza-se o ponto-e-vírgula após a definição da constante, pois é um comando de pré-processador
- Não utiliza-se o operador de atribuição (=)

Constantes

```
4  int main()  
5  {  
6      #define NUM 6  
7  
8      int pontos = NUM;  
9  
10     return 0;  
11 }
```

Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis
- 4 Operadores Aritméticos**
- 5 Operadores Relacionais e Lógicos
- 6 Comandos de Entrada e Saída

Operadores Aritméticos

Operação	Operador	Operador em C	Expressão
Adição	+	+	$n1 + n2$
Subtração	-	-	$n1 - n2$
Multiplicação	\times	*	$n1 * n2$
Divisão	\div	/	$n1 / 2$
Divisão inteira	div	/	$n1 / 2$
Resto	mod	%	$n1 \% 2$

Operadores Aritméticos

EXEMPLOS:

- $5 / 2$ resulta em 2, se 5 for um **int**
- $5 / 2$ resulta em 2.5, se 5 for um **float**
- $5 \% 2$ resulta em 1

Operadores Aritméticos

Precedência dos operadores

Ordem	Operador	Operações
1º	()	parênteses
2º	*, / ou %	divisão, multiplicação e resto
3º	+ ou -	soma e subtração

Cálculos da esquerda para a direita

Operadores Aritméticos

Álgebra: $m = \frac{(a + b + c + d + e)}{5}$

C: `m = (a + b + c + d + e) / 5;`

Figura: Equação na álgebra e sua representação em C.

Operadores aritméticos de atribuição

- Também chamados de operadores reduzidos
- Compilada mais rapidamente
- Grande diferença de desempenho do programa em situações com muitas iterações

6	<code>int n = 2;</code>
7	
8	<code>n = n + 5;</code>
9	
10	<code>n += 5;</code>

Operadores aritméticos de atribuição

6	<code>int n = 2;</code>
7	
8	<code>n = n - 5;</code>
9	<code>n -= 5;</code>
10	
11	<code>n = n * 3;</code>
12	<code>n *= 3;</code>
13	
14	<code>n = n / 2;</code>
15	<code>n /= 2;</code>
16	
17	<code>n = n % 4;</code>
18	<code>n %= 4;</code>

Operadores de incremento e decremento

- Operadores unários
- Forma simplificada de incrementar e decrementar em uma unidade uma variável
 - ++ Incremento
 - Decremento

Operadores de incremento e decremento

Oper	Exp	Como usar
++	++a	Incrementa a e usa o novo valor de a na expressão onde se localiza
	a++	Usa o valor atual de a na expressão onde se localiza e depois incrementa a
--	--b	Decrementa b e usa o novo valor de b na expressão onde se localiza
	b--	Usa o valor atual de b na expressão onde se localiza e depois decrementa b

Operadores de incremento e decremento

Observe os resultados das seguintes execuções de código:

```
1      #include <stdio.h>
2      #include <stdlib.h>
3
4      int main(){
5          int n = 5;
6
7          printf("Valor de n: %d\n", n++);
8          printf("Valor de n: %d\n", n);
9          system("pause");
10         return 0;
11     }
12
```


Operadores de incremento e decremento

Observe os resultados das seguintes execuções de código:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int n = 5;
6
7      printf("Valor de n: %d\n", ++n);
8      printf("Valor de n: %d\n", n);
9      system("pause");
10     return 0;
11 }
12
```

Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis
- 4 Operadores Aritméticos
- 5 Operadores Relacionais e Lógicos**
- 6 Comandos de Entrada e Saída

Operadores Relacionais

Operador matemático	Operador em C	Como usar
$=$	<code>==</code>	$x == y$
\neq	<code>!=</code>	$x != y$
$>$	<code>></code>	$x > y$
$<$	<code><</code>	$x < y$
\geq	<code>>=</code>	$x \geq y$
\leq	<code><=</code>	$x \leq y$

Operadores Lógicos

Operador matemático	Operador em C	Como usar
E (and)	&&	$p \ \&\& \ q$
OU (or)		$p \ \ q$
NÃO (not)	!	$!p$

Operadores Relacionais e Lógicos

Precedência dos operadores

Operador	Precedência
()	Maior precedência
!	
* / %	
+ -	
< <= > >=	
== !=	
&&	
=	Menor precedência

Súmario

- 1 Introdução
- 2 Programa em C
- 3 Variáveis
- 4 Operadores Aritméticos
- 5 Operadores Relacionais e Lógicos
- 6 Comandos de Entrada e Saída**

Comandos de Entrada e Saída

printf()

- Comando para saída de dados
- Imprime um texto no console (terminal)
- Existente na biblioteca `<stdio.h>`

Comandos de Entrada e Saída

printf()

- O conteúdo a ser impresso deverá estar entre aspas duplas: "conteúdo".
- **Sequência de escape** são combinações de caracteres objetivando ações na formatação da saída
- A \ é chamada de **caractere de escape**

Comandos de Entrada e Saída

Sequências de escape em C

Sequência de escape	Descrição
<code>\n</code>	Nova linha
<code>\t</code>	Tabulação horizontal
<code>\\</code>	Barra invertida
<code>\"</code>	Aspas duplas
<code>\'</code>	Aspa simples

Comandos de Entrada e Saída

EXEMPLO:

```
4  int main()  
5  {  
6      printf("Ola Mundo!\n");  
7  
8      return 0;  
9  }
```

Saída no console:

Olá mundo!

Comandos de Entrada e Saída

EXEMPLO:

```
int main()
{
    printf("Bom dia!!! \nTudo bem?");

    return 0;
}
```

Saída no console:

```
Bom dia!!!
Tudo bem?
```

O programa escreve no console o texto **Bom dia!!!** em seguida passa para a próxima linha (`\n`) e então escreve o texto **Tudo bem?**

Comandos de Entrada e Saída

EXEMPLO:

```
int main()  
{  
    printf("Bom dia!!! \n Tudo bem");  
    return 0;  
}
```

Saída no console:

```
Bom dia!!!  
Tudo bem?
```

O programa escreve no console o texto **Bom dia!!!** em seguida passa para a próxima linha (`\n`), adiciona um espaço em branco e escreve o texto **Tudo bem?**

Comandos de Entrada e Saída

EXEMPLO:

```
int main()  
{  
    printf("Bom dia!!! \n\nTudo bem?");  
  
    return 0;  
}
```

Saída no console:

Bom dia!!!

Tudo bem?

O programa escreve no console o texto **Bom dia!!!** em seguida passa para a próxima linha (`\n`), salta outra linha (`\n`) e escreve o texto **Tudo bem?**

Comandos de Entrada e Saída

`printf()`

Para imprimir o conteúdo de variáveis deve-se usar o comando em duas partes

- **1ª parte:** texto como será impresso, formatado com caracteres especiais e especificadores de conversão
- **2ª parte:** as variáveis

```
printf("%d ", variavel);
```

Comandos de Entrada e Saída

Especificadores de conversão

Tipo de dado	Especificador
long double	%Lf
double	%f
float	%f
unsigned long int	%lu
long int	%ld
int	%i ou %d
short	%hd
char	%c
cadeia de caractere	%s

As duas partes do printf()

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int n = 3;
7
8      printf("valor de n: %d, dobro de n: %d", n, 2*n);
9
10     return 0;
11 }
```

Parte 1

Parte 2

valor de n: 3, dobro de n: 6

Comandos de Entrada e Saída

PRÁTICA: Escreva o código em C usando o comando `printf` para escrever os seguintes textos no console:

- 1- Estudarei hoje
Amanha irei para festa
- 2- Se eu não estudar para a prova
A prova vai ser difícil
- 3- `int nota = 9;` //variável do programa
Minha nota da prova foi ...
O dobro da minha nota é ...
A metade da minha nota é ...

Comandos de Entrada e Saída

scanf()

- Existente na biblioteca `<stdio.h>`
- Comando para entrada de dados pelo dispositivo padrão, que normalmente é o teclado
- Usuário escreve os dados pelo teclado, e o valor digitado pelo usuário após pressionar a tecla **Enter** é atribuído para uma variável especificada
- Usa os especificadores de conversão

Comandos de Entrada e Saída

scanf()

- Formado por duas partes
- **1ª parte:** texto formatado com especificadores de conversão
- **2ª parte:** variáveis que receberão os dados de entrada do usuário são precedidas pelo **&**
- O **&** é o operador de memória indicando o local da memória onde a variável está

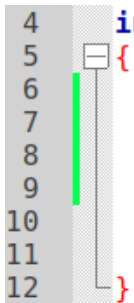
scanf()

```
1  #include <stdio.h>|
2  #include <stdlib.h>
3
4  int main()
5  {
6      int n;
7      Parte 1
8      scanf("%d", &n);
9              Parte 2
10     return 0;
11 }
12
```

O programa espera o usuário digitar um valor e pressionar a tecla **ENTER** e então atribui o valor digitado pelo usuário para a variável inteira. **n**

scanf()

É possível ler duas ou mais variáveis em um mesmo **scanf**, bastando adicionar novos especificadores de conversão e variáveis que receberão os valores.



```
4 int main()  
5 {  
6     int x, y;  
7     x = y = 0;  
8  
9     scanf("%i%i", &x, &y);  
10  
11     return 0;  
12 }
```