



UNIVERSIDADE
FEDERAL DO CARIRI

Introdução a C

Professora Dra. Luana Batista da Cruz
luana.batista@ufca.edu.br

Roteiro

01 Introdução

02 Linguagem C

01

Introdução

Histórico - Linguagem C

Por que aprender C?

Características da linguagem C

Linguagem compilada e interpretada

Fluxo do compilador C

Ambiente de desenvolvimento

Histórico - Linguagem C

- Foi criada por Dennis Ritchie em 1972 no centro de pesquisas da Bell Laboratories
- Seu primeiro uso importante foi a reescrita do Sistema Operacional Unix, que até então era escrito em assembly
- Se tornou tão popular que por volta de 1980, já existiam várias versões de compiladores C oferecidas por várias empresas
- É uma linguagem de uso geral. O melhor uso dela é feito em programas que lidam diretamente com hardware, como um sistema operacional ou um driver

Por que aprender a C?

- É uma ótima linguagem para programadores iniciantes
- Depois de aprender C, terá muitas semelhanças com outras linguagens de programação (Java, Javascript, Shell, PHP, etc). Além de todas as linguagens que são C-alguma coisa (C++, C#, Objective-C, etc)
- Principais linguagens do mercado/comerciais são baseadas em C
- Sistemas operacionais (Linux e Unix) são escritos em C

Por que aprender a C?

- C é fundamental, para uma sólida formação em programação
- A linguagem C opera muito próxima ao hardware. Isto pode gerar uma dificuldade extra para o aprendizado. No entanto, a programação em C adquire uma boa compreensão de como o computador funciona
- Em C, é necessário gerenciar explicitamente a memória que é alocada. Pode-se manipular diretamente endereços de memória e precisa entender o conceito de passagem de parâmetro por valor e por referência (ponteiro)

Características da linguagem C

- A linguagem C pertence a uma família de linguagens cujas **características** são
 - **Modularidade:** um programa é dividido em vários blocos de programação distintos, ou seja, funções que não estão interligadas
 - **Recursos de baixo nível:** tem muitos recursos para controlar a memória da sua máquina
 - **Linguagem procedural:** especifica uma série de etapas e procedimentos bem estruturados dentro de seu contexto de programação para compor um programa
 - **Simplicidade:** a sintaxe é relativamente fácil de aprender e, seguindo as regras, dificilmente vai cometer erros que possam comprometer o seu programa

Características da linguagem C

- A linguagem C pertence a uma família de linguagens cujas **características** são
 - **Portabilidade:** é extremamente portátil, pois os programas escritos em C podem ser executados e compilados em qualquer sistema com nenhuma ou pequenas alterações
 - **Compilação:** seus programas são rapidamente compilados gerando um arquivo executável
 - **Uso geral:** embora não seja, obviamente, a melhor escolha para todas as aplicações, pode-se fazer quase tudo com C

Linguagem compilada e interpretada

- **Compilada**

- O compilador pode ser definido como um programa que traduz todo o código escrito em uma linguagem de programação (código-fonte) em um código de máquina, gerando arquivos adicionais que consigam ser executados pelo computador

- **Interpretada**

- O interpretador, ao contrário do compilador, NÃO traduz o código-fonte inteiro para depois executá-lo, já que essa conversão ocorre simultaneamente à execução do código, deixando de lado a necessidade da criação de arquivos adicionais em código de máquina para serem executados posteriormente

Linguagem compilada e interpretada

- **Compilada**

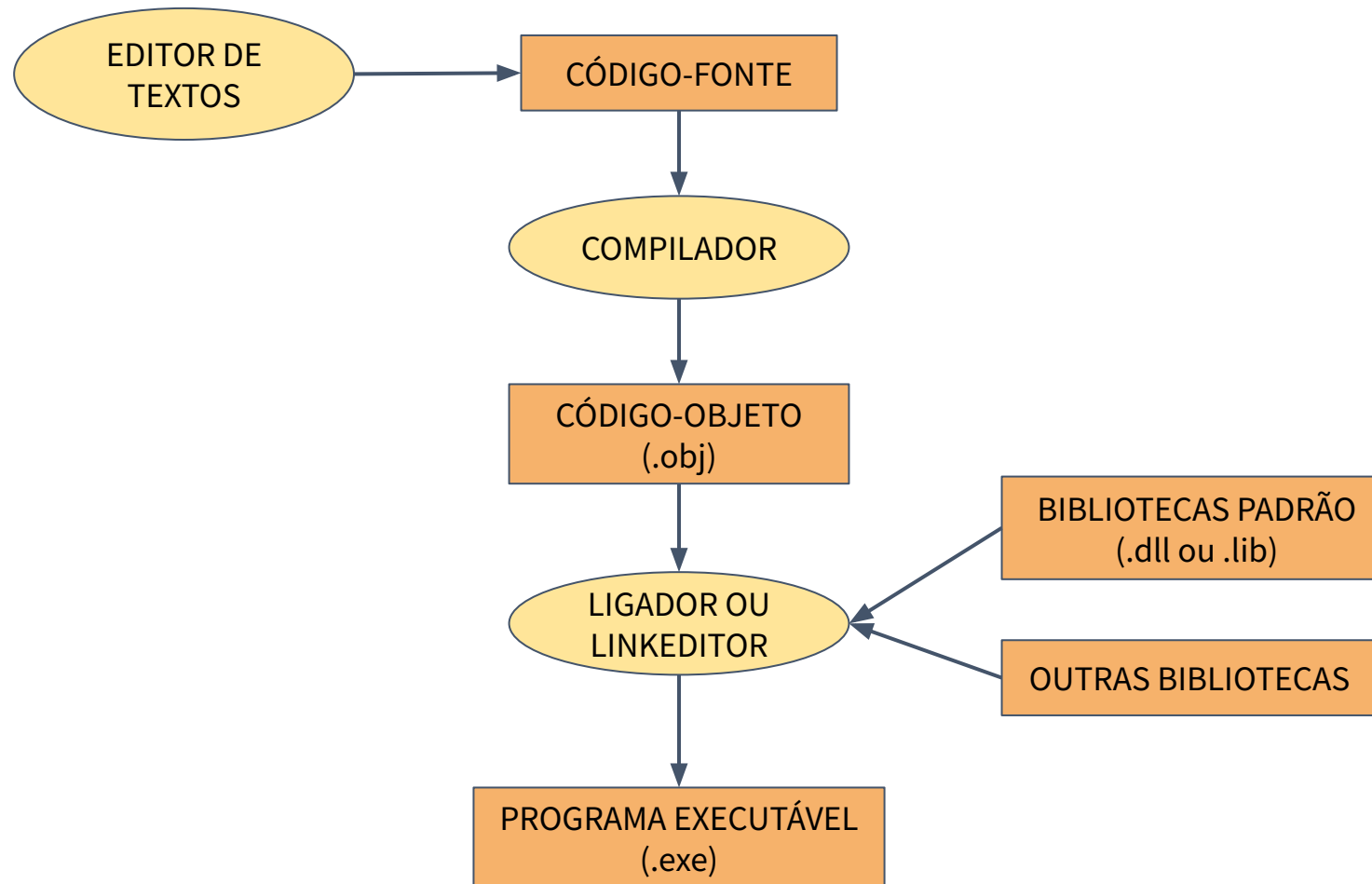
1. O programa conversor recebe a primeira instrução do programa fonte, verifica se está escrita corretamente
2. Se sim, converte para linguagem de máquina e passe para a próxima instrução, repetindo o processo sucessivamente até a última instrução do programa fonte
3. Se a transformação da última instrução do programa fonte foi concluída e nenhum erro foi detectado, o computador retorna à primeira instrução, já transformada em linguagem de máquina, e a executa
4. Passa à instrução seguinte, executa-a, etc., até a última

Linguagem compilada e interpretada

- **Interpretada**

1. O programa conversor recebe a primeira instrução do programa fonte, verifica se está escrita corretamente, converte-a em linguagem de máquina e, em seguida, informa ao computador para executar essa instrução
2. Depois repete o processo para a segunda instrução, e assim sucessivamente, até a última instrução do programa fonte
3. Quando a segunda instrução é trabalhada, a primeira é perdida, isto é, apenas uma instrução fica na memória em cada instante

Fluxo do compilador C



Fluxo do compilador C

- **Principais termos**

- **Código-fonte:** criado em um editor de textos, contendo os comandos da linguagem de programação (C, Pascal...). Serve como entrada para o compilador
- **Código-objeto:** criado pela conversão do código-fonte em linguagem de máquina. É gerado pelo compilador. Só é criado quando não há erros no código-fonte. (extensão do código-objeto: .OBJ)
- **Ligador ou Linkeditor:** "junta" o código-objeto com as bibliotecas necessárias para gerar o programa-executável. (extensões das bibliotecas: .DLL ou .LIB)
- **Programa executável:** código que pode ser executado pelo sistema operacional. (extensão do programa-executável: .EXE)

Ambiente de desenvolvimento

- Ou IDE (Integrated Development Environment)
- Normalmente incluem
 - Um editor de texto para códigos-fonte
 - Complementação automática de código (code completion)
 - Destacamento de sintaxe através de cores (syntax highlighting)
 - Um compilador (compiler)
 - Um depurador (debugger): permite executar o programa passo-a-passo

Ambiente de desenvolvimento

- IDEs gratuitas para C/C++ recomendadas para o curso
 - Dev C++ - <http://sourceforge.net/projects/dev-cpp/>
 - Code::Blocks - <http://www.codeblocks.org/>
- Aplicativos para fazer programação na plataforma Android
 - <https://www.thecrazyprogrammer.com/2015/05/5-best-apps-to-do-programming-on-android-platform.html>
- Site
 - <https://replit.com/>

Linguagem C

Tipo básicos de variáveis

Modificadores de tipos

Operadores aritméticos básicos, relacionais e lógicos

Operações de fluxo (condição e repetição)

Linguagem C

- **Tipo básicos de variáveis (tipo de dados)**
 - **char**: o valor armazenado é um caractere. Caracteres geralmente são armazenados em códigos (usualmente o código ASCII)
 - **int**: número inteiro (positivos, negativos e o 0)
 - **float**: número em ponto flutuante de precisão simples. São conhecidos normalmente como números reais
 - **double**: número em ponto flutuante de precisão dupla
 - **void**: este tipo serve para indicar que um resultado não tem um tipo definido. Não retorna um valor

Linguagem C

- **Modificadores de tipos**

- Podem aumentar ou diminuir a capacidade de armazenamento e definir se a faixa numérica será a positiva ou então negativa
 - **signed**: números positivos e negativos
 - **unsigned**: números positivos
 - **long**: aumentar a capacidade de armazenamento
 - **short**: diminuir a capacidade de armazenamento

Linguagem C

- **Modificadores de tipos**

Tipo	Tamanho em bytes	Faixa mínima
char	1	-127 a 127
unsigned char	1	0 a 255
int	4	-2.147.483.648 a 2.147.483.647
unsigned 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	-4.294.967.295 a 4.294.967.295
unsigned long int	4	0 a 4.294.967.295
float	4	Seis dígitos de precisão
double	8	Dez dígitos de precisão
long double	10	Dez dígitos de precisão

Linguagem C

- **Variáveis**

- **Regras básicas para nomear variáveis**

- Todo nome só pode conter letras e/ou dígitos
 - Apenas o caractere símbolo "_" pode ser usado
 - Todo primeiro caractere deve ser sempre uma letra
 - Letras maiúsculas e minúsculas são consideradas caracteres diferentes

- **Declaração de variáveis**

- `int i, idade, numero;`
 - `float salario, altura;`
 - `unsigned char sexo, letra;`

Linguagem C

- **Variáveis**

- **Regras básicas para nomear variáveis**

- Todo nome só pode conter letras e/ou dígitos
 - Apenas o caractere símbolo "_" pode ser usado
 - Todo primeiro caractere deve ser sempre uma letra
 - Letras maiúsculas e minúsculas são consideradas caracteres diferentes

- **Declaração de variáveis**

- `int i, idade, numero;`
 - `float salario, altura;`
 - `unsigned char sexo, letra;`

Obs.: não só as variáveis mas toda a linguagem C é “**Case Sensitive**”, isto é, maiúsculas e minúsculas fazem diferença. Por exemplo: **Idade** ≠ **idade**, ou seja, são duas variáveis diferentes

Linguagem C

- **Variáveis**

- **Regras básicas para nomear variáveis**

- Todo nome só pode conter letras e/ou dígitos
 - Apenas o caractere símbolo "_" pode ser usado
 - Todo primeiro caractere deve ser sempre uma letra
 - Letras maiúsculas e minúsculas são consideradas caracteres diferentes

- **Declaração de variáveis**

- `int i, idade, numero;`
 - `float salario, altura;`
 - `unsigned char sexo, letra;`

; após a declaração

Obs.: não só as variáveis mas toda a linguagem C é “**Case Sensitive**”, isto é, maiúsculas e minúsculas fazem diferença. Por exemplo: **Idade** ≠ **idade**, ou seja, são duas variáveis diferentes

Linguagem C

- **Variáveis**

- **Booleanas**

- A linguagem C não possui explicitamente variáveis do tipo booleano. Entretanto, a linguagem considera um número com valor 0 (zero) igual a falso e qualquer número **diferente de 0 (zero) igual a verdadeiro**


Linguagem C

- **Variáveis**

- **Atribuição**

- A atribuição é realizada usando o símbolo “=”
 - idade = 31;
 - sexo = 'm';

Caracteres usam
aspas simples



Linguagem C

- **Operadores aritméticos básicos**

Operador	Símbolo	Exemplo
Adição	+	$a + b$
Subtração	-	$a - b$
Multiplicação	*	$a * b$
Divisão	/	a / b
Resto de Divisão Inteira	%	$a \% b$

Linguagem C

- Operadores relacionais e lógicos

Op. relacionais são usados para fazer comparações entre variáveis. O resultado é um valor booleano (verdadeiro ou falso)

Op. lógicos são usados quando é necessário usar duas ou mais condições dentro da mesma **instrução if** para que seja tomada uma única decisão cujo resultado será verdadeiro ou falso

Operador	Símbolo	Exemplo
Igual	==	a == b
Diferente	!=	a != b
Maior	>	a > b
Maior ou igual	>=	a ≥ b
Menor	<	a < b
Menor ou igual	<=	a ≤ b
Conjunção (e)	&&	a && b
Disjunção (ou)		a b
Negação	!	! c

Linguagem C

- Operadores relacionais e lógicos

Operador	Símbolo	Exemplo	Relacionais
Igual	==	$a == b$	
Diferente	!=	$a != b$	
Maior	>	$a > b$	
Maior ou igual	>=	$a \geq b$	
Menor	<	$a < b$	
Menor ou igual	<=	$a \leq b$	Lógicos
Conjunção (e)	&&	$a \&\& b$	
Disjunção (ou)		$a b$	
Negação	!	$!c$	

Linguagem C

- **Comando de saída (printf)**

- `printf (<info. de controle>, <lista de variáveis>);`
- Informações de controle
 - É uma descrição do que vai aparecer na tela. Também é a definição do tipo de dado do valor a ser exibido (geralmente de uma variável). Isto é feito usando-se os códigos de controle, que usam a notação %

Código	Significado
%d	Inteiro
%f	Float
%c	Caractere
%s	String
%%	Coloca na tela um %

Linguagem C

- **Comando de saída (printf)**


- Exemplos

- `printf ("%f", 40.345)`
 - "40.345"
 - `printf ("Um caractere %c e um inteiro %d", 'D', 120)`
 - "Um caractere D e um inteiro 120"
 - `printf ("%s eh um exemplo", "Este")`
 - "Este eh um exemplo"
 - `printf ("%s%d%%", "Juros de ", 10)`
 - "Juros de 10%"

Linguagem C

- **Comando de entrada (scanf)**

- `scanf (<info. de controle>, &<lista de variáveis>);`
- Exemplos
 - `scanf ("%f", &salario);`
 - `scanf ("%d", &idade);`
 - `scanf ("%c", &letra);`
 - `scanf ("%d %f %c", &idade, &salario, &letra);`
- O caractere **&** indica que o valor será armazenado no endereço de memória da variável



Para a variável **salario** vai atribuir o valor do **tipo float**

Linguagem C

- **Caracteres de escape**

Caractere	Significado
\a	Aviso sonoro
\n	Nova linha
\t	Tabulação horizontal
\v	Tabulação vertical
\\	Caractere de barra invertida
\'	Apóstrofe
\"	Aspas
\?	Interrogação

Linguagem C

- **Abreviação de expressões**

- A linguagem C admite as seguintes equivalências, que podem ser usadas para simplificar expressões ou para facilitar o entendimento de um programa

Expressão original	Expressão equivalente
$x = x + k;$	$x += k;$
$x = x - k;$	$x -= k;$
$x = x * k;$	$x *= k;$
$x = x / k;$	$x /= k;$
$x = x + 1$	$x++$
	$++x$
$x = x - 1$	$x--$
	$--x$

Exemplo em C (estrutura básica)

int indica que a função **main** retorna um valor do tipo inteiro

#include inclui a biblioteca **stdio.h**. Essa biblioteca possui declarações de funções de I/O

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

Os caracteres chave { e } delimitam o início e fim da função **main**, respectivamente

A função **main** será a primeira a ser chamada quando o programa for executado

return retorna um valor da função **main**

Exemplo em C (estrutura básica)

- **Comentários**

- Tipos de comentários
 - Comentário de uma linha: //
 - Comentário de múltiplas linhas: /* */

```
1  #include <stdio.h>
2
3  int main (){
4      //Comentário de uma linha
5      printf("Oi mundo...\n");
6      /*
7          Comentário de múltiplas
8          linhas
9      */
10     return 0;
11 }
```

Exemplo em C

- Inserindo **system("PAUSE")** para fazer o programa “**parar**”

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

O arquivo **stdlib.h** possui funções de alocação de memória, controle de processos, conversões e outras

“Para” a execução do programa

Exemplo em C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      printf ("Teste %% %%\n");
6      printf ("%f\n",40.345);
7      printf ("Um caractere %c e um inteiro %d\n",'D',120);
8      printf ("%s eh um exemplo\n","Este");
9      printf ("%s%d%%\n","Juros de ",10);
10
11     system ("PAUSE");
12     return 0;
13 }
```

Qual a saída?

Exemplo em C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      printf ("Teste %% %%\n");
6      printf ("%f\n",40.345);
7      printf ("Um caractere %c e um inteiro %d\n",'D',120);
8      printf ("%s eh um exemplo\n","Este");
9      printf ("%s%d%%\n","Juros de ",10);
10
11     system ("PAUSE");
12     return 0;
13 }
```

Qual a saída?

Teste % %

40.345000

Um caractere D e um inteiro 120

Este eh um exemplo

Juros de 10%

Linguagem C

- **Exercício 1:** dado dois números, calcule a multiplicação, divisão, soma e subtração

Linguagem C

- **Exercício 1:** dado dois números, calcule a multiplicação, divisão, soma e subtração

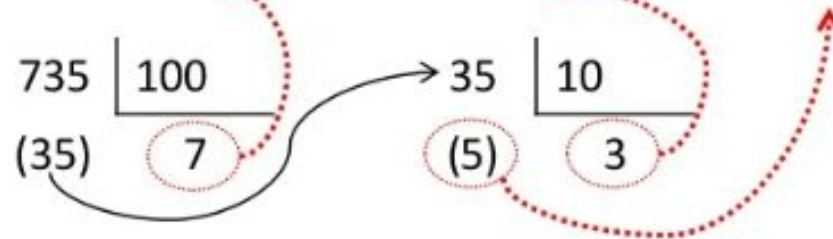
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      float n1, n2, soma, sub, div, mult;
6      printf("Digite o primeiro valor: ");
7      scanf("%f", &n1);
8      printf("Digite o segundo valor: ");
9      scanf("%f", &n2);
10
11     soma = n1 + n2;
12     sub = n1 - n2;
13     div = n1 / n2;
14     mult = n1 * n2;
15
16     printf("Soma: %f\n", soma);
17     printf("Subtracao: %f\n", sub);
18     printf("Divisao: %f\n", div);
19     printf("Multiplicao: %f\n", mult);
20
21     system("PAUSE");
22     return 0;
23 }
```

Linguagem C

- **Exercício 2:** dado um número inteiro de três algarismos, exibir cada algarismo separadamente

Entrada: 735

Saída: Centenas = 7 Dezenas = 3 Unidade = 5



Centenas:

$$735 / 100 = 7$$

Dezenas:

$$(735 \% 100) / 10 = 3$$

Unidade:

$$(735 \% 100) \% 10 = 5$$

Linguagem C

- **Exercício 2:** dado um número inteiro de três algarismos, exibir cada algarismo separadamente

```
1  ✓ #include <stdio.h>
2    #include <stdlib.h>
3
4  ✓ int main() {
5      int n1, centena, dezena, unidade;
6
7      printf("Digite um numero: ");
8      scanf("%d", &n1);
9
10     centena = n1 / 100;
11     dezena = (n1 % 100) / 10;
12     unidade = (n1 % 100) % 10;
13
14     printf("Centena: %d\n", centena);
15     printf("Dezena: %d\n", dezena);
16     printf("Unidade: %d\n", unidade);
17
18     system("PAUSE");
19     return 0;
20 }
```

Linguagem C

- **Operações de fluxo (sentença de condição)**
 - Comando **se**
 - Altera o fluxo de execução de um programa baseado no valor (verdadeiro ou falso) de uma condição

```
if (<condição>)  
    <comandos a serem executadas caso a condição resulte em VERDADEIRO>;  
[ else  
    <comandos a serem executadas caso a condição resulte em FALSO>; ]
```

Linguagem C

- **Comando se**

- **Exemplo:** o usuário deve informar sua idade. Posteriormente, verifique se o usuário é de maior

Linguagem C

- **Comando se**

- **Exemplo:** o usuário deve informar sua idade. Posteriormente, verifique se o usuário é de maior

```
1  ∨ #include <stdio.h>
2    #include <stdlib.h>
3
4  ∨ int main(){
5      int idade;
6      printf ("Digite um número:");
7      scanf ("%d", &idade);
8      if (idade >= 18)
9          printf ("de maior\n");
10     else
11         printf ("de menor\n");
12
13     system ("PAUSE");
14     return 0;
15 }
```

Linguagem C

- **Comando se**

- **Exercício 1:** leia um número inteiro e verifique se é par ou ímpar. Se for par, imprima o valor multiplicado por 2, caso contrário, o valor adicionado mais 3

Linguagem C

- **Comando se**

- **Exercício 1:** leia um número inteiro e verifique se é par ou ímpar. Se for par, imprima o valor multiplicado por 2, caso contrário, o valor adicionado mais 3

```
1  ✓ #include <stdio.h>
2    #include <stdlib.h>
3
4  ✓ int main() {
5      int n1;
6
7      printf("Digite um numero: ");
8      scanf("%d", &n1);
9
10  ✓ if(n1 % 2 == 0){
11      | printf("%d", n1*2);
12  }else
13      | printf("%d", n1+3);
14
15      system("PAUSE");
16      return 0;
17  }
```

Linguagem C

- Operações de fluxo (sentença de **repetição**)
 - Comando **for**
 - O comando **for** permite que um certo trecho de programa seja executado um **determinado número de vezes**

```
for (<variável>; <condição>; <atualizar-variável>)  
    <sequência-de-comandos>;
```

Linguagem C

- **Exemplo:** imprima os números de um a quatro usando o **for**

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

Se o **for** tiver mais de um comando é necessário a utilização dos caracteres de chaves para marcar o início e fim do comando: { (início) e } (fim)

Linguagem C

- **Comando for**

- **Exercício 1:** faça um algoritmo que apresente os números múltiplos de 2 e 3 no intervalo de 1 à 100

Linguagem C

- **Comando for**

- **Exercício 1:** faça um algoritmo que apresente os números múltiplos de 2 e 3 no intervalo de 1 à 100

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int i, n1;
6
7      for(i=1;i<=100;i++){
8          if((i % 2 == 0) && (i % 3 == 0))
9              printf("%d\n", i);
10     }
11
12     system("PAUSE");
13     return 0;
14 }
```

Linguagem C

- Operações de fluxo (sentença de **repetição**)
 - Comando **while**
 - Repete uma sequência de comandos **ENQUANTO** uma determinada condição for satisfeita

```
while (<condição>
    <sequência-de-comandos>;
```

Linguagem C

- **Exemplo:** imprima os números de um a quatro usando o **while**

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main (){
5      int i = 1;
6      while(i <=4) {
7          printf ("Numero = %d\n", i);
8          i = i + 1;
9      }
10
11     system ("PAUSE");
12     return (0);
13 }
```

Linguagem C

- **Comando while**
 - **Exercício 1:** faça um algoritmo que realize a soma de cinco valores fornecidos pelo usuário

Linguagem C

- **Comando while**

- **Exercício 1:** faça um algoritmo que realize a soma de cinco valores fornecidos pelo usuário

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int i = 0;
6      float n, soma = 0;
7
8      while(i < 5){
9          printf("Digite um numero: ");
10         scanf("%f", &n);
11         soma = soma + n;
12         i++;
13     }
14
15     printf("Resultado da soma de valores: %.2f", soma);
16
17     system("PAUSE");
18     return 0;
19 }
```

Linguagem C

- Operações de fluxo (sentença de **repetição**)
 - Comando **do while**
 - O comando **do while** permite que um certo trecho de programa seja executado **ENQUANTO** uma certa condição for verdadeira

```
do {  
    <sequência-de-comandos>;  
} while (<condição>;
```

Linguagem C

- Operações de fluxo (sentença de **repetição**)
 - Comando **do while**
 - O comando **do while** permite que um certo trecho de programa seja executado **ENQUANTO** uma certa condição for verdadeira

```
do {  
    <sequência-de-comandos>;  
} while (<condição>;
```

IMPORTANTE!

Se **if**, **else**, **while**, **for** tiverem mais de um comando é necessário a utilização dos caracteres de chaves para marcar o início e fim do comando: { (início) e } (fim)

Linguagem C

- **Exemplo:** imprima os números de um a quatro usando o **do while**

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main (){
5      int i = 1;
6      do {
7          printf ("Numero = %d\n", i);
8          i = i + 1;
9      } while(i <=4);
10
11     system ("PAUSE");
12     return (0);
13 }
```

A utilização dos caracteres de chaves { e } são obrigatórios no comando **do while**

Linguagem C

- **Comando do while**

- **Exercício 1:** crie um algoritmo que faça a leitura de vários números digitados pelo usuário e apresente a quantidade de valores positivos. O algoritmo encerra quando for digitado o valor zero

Linguagem C

- **Comando do while**

- **Exercício 1:** crie um algoritmo que faça a leitura de vários números digitados pelo usuário e apresente a quantidade de valores positivos. O algoritmo encerra quando for digitado o valor zero

```
1  ✓ #include <stdio.h>
2    #include <stdlib.h>
3
4  ✓ int main() {
5      int n, cont_pos = 0;
6
7  ✓  do{
8      printf("Digite um valor: ");
9      scanf("%d", &n);
10  ✓  if(n > 0){
11      cont_pos = cont_pos + 1;
12      }
13  }while(n != 0);
14
15  printf("Quantidade de valores positivos: %d", cont_pos);
16
17  system("PAUSE");
18  return 0;
19  }
```

Linguagem C

- **Comando break**
 - Pode quebrar a execução de um comando (como no caso do switch) ou interromper a execução de qualquer loop. O break faz com que a execução do programa continue na primeira linha seguinte ao loop ou bloco que está sendo interrompido

Linguagem C

- Comando break

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      int i;
6      i = 1;
7      for(i; i <= 20; i++) {
8          if(i % 2 == 0){
9              break;
10         }
11     }
12
13     system ("PAUSE");
14     return 0;
15 }
```

Linguagem C

- **Comando de seleção (switch)**
 - O conteúdo de uma variável é **comparado** com um valor constante, e caso a comparação seja verdadeira, **um determinado comando é executado**

Linguagem C

- Comando de seleção (switch)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main () {
4      int num;
5      printf ("Digite um numero: ");
6      scanf ("%d",&num);
7      switch (num) {
8          case 9: printf ("\n\nO numero eh igual a 9.\n");
9                  break;
10         case 10: printf ("\n\nO numero eh igual a 10.\n");
11                 break;
12         case 11: printf ("\n\nO numero eh igual a 11.\n");
13                 break;
14         default: printf ("\n\nO numero nao eh nem 9 nem 10 nem 11.\n");
15     }
16     system ("PAUSE");
17     return (0);
18 }
```

Default (opcional): é executado se nenhuma coincidência for detectada

Linguagem C

- **Caracteres**
 - Algoritmo para listar a tabela ASCII

```
1  √ #include <stdio.h>
2    #include <stdlib.h>
3
4  √ int main() {
5      unsigned char simbolo = 0;
6  √    while(simbolo < 255){
7          printf("%c=%d  ", simbolo, simbolo);
8          simbolo = simbolo + 1;
9      }
10     system ("PAUSE");
11     return 0;
12 }
```


Linguagem C

- **Caracteres**

- Funções de entrada para caracteres
 - **getch()**: apenas retorna o caractere pressionado sem mostrá-lo na tela

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4
5  int main(){
6      char Ch;
7      Ch = getch();
8      printf ("Tecla = %c\n",Ch);
9
10     system ("PAUSE");
11     return 0;
12 }
```

conio.h são úteis para manipular caracteres na tela, especificar cor de carácter e de fundo

Linguagem C

- **Caracteres**

- Funções de entrada para caracteres
 - **getche()**: mostra o caractere na tela antes de retorná-lo

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4
5  int main(){
6      char Ch;
7      Ch = getche();
8      printf ("\nTecla = %c\n",Ch);
9
10     system ("PAUSE");
11     return 0;
12 }
```

Linguagem C

- **Caracteres**

- Em muitos casos, ao ler um caractere/string pode encontrar alguns problemas
 - Toda a informação que digitamos no teclado é armazenada em um buffer e fica disponível para nossa utilização
 - Quando usamos a função `scanf()`, ela recupera a informação do buffer. Porém, ela pode deixar "sujeira" no buffer, comprometendo futuras leituras

Linguagem C

- **Caracteres**
 - Entrada comum de caracteres

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      char caractere_1, caractere_2;
6
7      printf("Digite um caractere: ");
8      scanf("%c", &caractere_1);
9
10     printf("Digite outro caractere: ");
11     scanf("%c", &caractere_2);
12
13     system("PAUSE");
14     return 0;
15 }
```

Digite um caractere: a
Digite outro caractere: Pressione qualquer tecla para continuar. . .

Linguagem C

- **Caracteres**
 - Entrada comum de caracteres

```
1  ✓ #include <stdio.h>
2    #include <stdlib.h>
3
4  ✓ int main() {
5      char caractere_1, caractere_2;
6
7      printf("Digite um caractere: ");
8      scanf("%c", &caractere_1);
9      setbuf(stdin, NULL);
10
11     printf("Digite outro caractere: ");
12     scanf("%c", &caractere_2);
13     setbuf(stdin, NULL);
14
15     system("PAUSE");
16     return 0;
17 }
```

Limpa o buffer

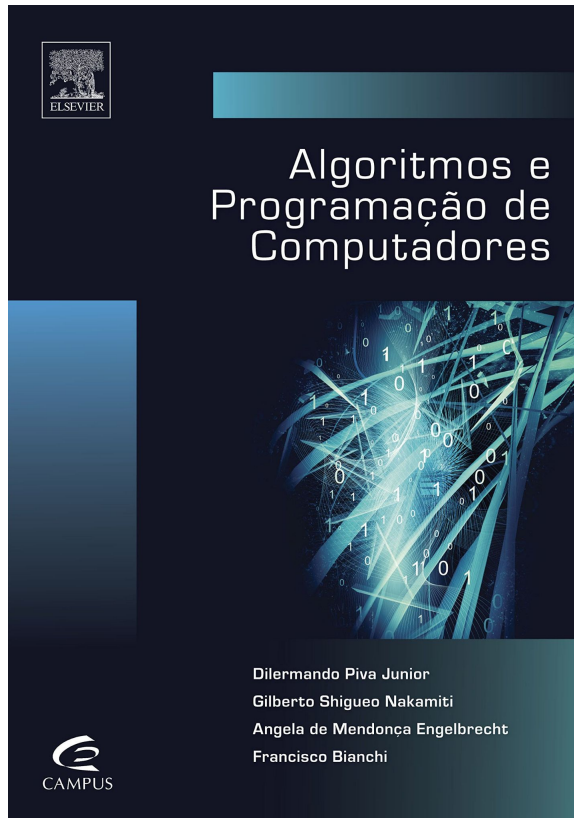
```
Digite um caractere: a
Digite outro caractere: b
Pressione qualquer tecla para continuar. . .
```

Resumindo..

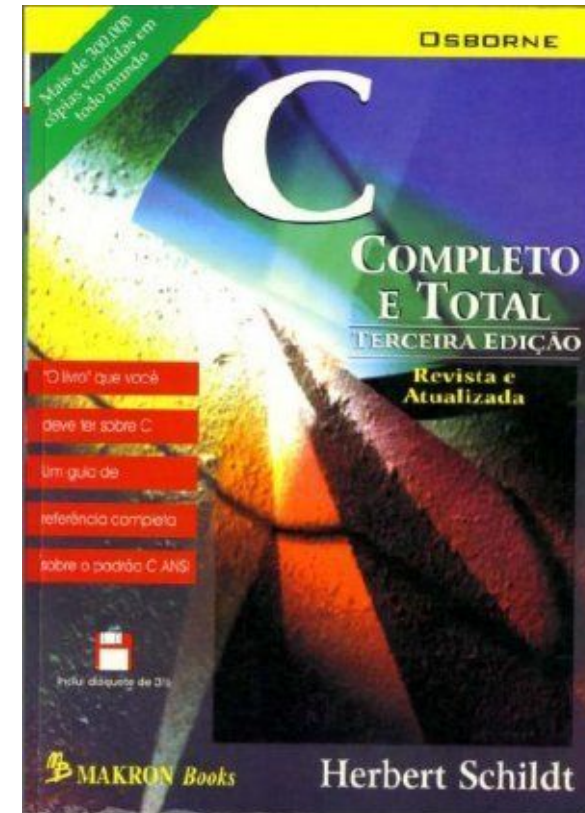
- Histórico da linguagem C
- Principais características da linguagem C
- Linguagem compilada e interpretada
- Linguagem C (conceitos e aplicações)



Referências



PIVA, D. J. et al. **Algoritmos e programação de computadores**. Rio de Janeiro, RJ: Elsevier, 2012.



SCHILDT, Herbert. **C completo e total**. Makron, 1997.