Algoritmos e Programação de Computadores

Linguagem e Técnica de Programação - C

* *Todos os direitos reservados nos termos da Lei Nº 9.610/98.*

Programa da Disciplina

- **Organização básica de computadores**
- **Algoritmos**
- **Introdução a linguagem de programação estruturada**
- **Tipos de dados, identificadores, constantes e variáveis**
- **Comandos de entrada/saída e operadores**
- **Estruturas de decisão (*if*, *switch*)**
- **Estruturas de repetição (*for*, *while*, *do-while*)**
- **Avaliação P1**
- **Vetores e matrizes**
- ***Strings* e estruturas**
- **Modularização de programas**
- **Ponteiros e alocação dinâmica de memória**
- **Avaliação P2**

Sumário

Comandos de Entrada / Saída e Operadores

- **Saída - *printf* (), *putchar* ()**
- **Entrada - *scanf* (), *getchar* ()**
- **Operador de atribuição (=)**
- **Operadores aritméticos (+, -, *, /, %)**
- **Operadores aritméticos de atribuição (+=, -=, *=, /=, %=)**
- **Operadores de incremento / decremento (++, -- pré e pós-fixado)**
- **Operadores relacionais (==, !=, >, >=, <, <=)**
- **Operadores lógicos (&&, ||, !)**
- **Precedência de operadores**

A função *printf*()

- **Função para impressão de dados formatados na saída padrão (*stdout* - tela).**
- **Definida na biblioteca padrão de C (stdio.h).**
- **Permite o uso de caracteres especiais e códigos de formatação.**

Sintaxe: printf ("<texto para impressão>", <expressão1>, <expressão2>,...);

- **texto para impressão = texto que será impresso na tela, podendo incluir caracteres especiais e códigos de formatação.**
- **expressão = argumento opcional que indica um valor que será impresso em texto para impressão. Pode ser uma constante, variável, expressão ou chamada a outra função.**

A função *printf*() – Caracteres Especiais

Caracter	Efeito
\a	Soa o alarme do computador
\b	O cursor retrocede (backspace)
\f	Alimenta página na impressora
\n	O cursor avança para uma nova linha
\r	O cursor retrocede para o início da linha
\t	Tabulação horizontal
\v	Tabulação vertical
\”	Exibe aspas dupla
\’	Exibe aspas simples (apóstrofo)
\\	Exibe uma barra invertida

A função *printf*() – Códigos de Formatação

Código	Formato
%c	Caractere simples
%s	Cadeia de caracteres (*string*)
%d	Número inteiro com sinal
%u	Número inteiro sem sinal
%ld	Número inteiro longo com sinal
%f	Número em ponto flutuante
%lf	Número em ponto flutuante longo (double)
%e	Número em notação científica
%o	Número octal (base 8)
%x	Número hexadecimal (base 16)
%p	Valor armazenado em um ponteiro (endereço)

A função *printf*() – Códigos de Formatação

Permitem também controlar:

- **O tamanho mínimo para impressão de um campo;**
- **A precisão e o arredondamento em campos de ponto flutuante;**
- **O alinhamento à direita ou à esquerda;**
- **O preenchimento com zeros à esquerda.**

A função *printf*() – Constante

```
1.  #include <stdio.h>
2.  #define UNIVERSIDADE "Unicamp"
3.  int main()
4.  {
5.      printf ("Você está na %s", UNIVERSIDADE);
6.      return 0;
7.  }
```

```
Você está na Unicamp
```

A função *printf*() – Variáveis e Expressão

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int nota1 = 8, nota2 = 10;
5.      printf ("A sua média é: %d", (nota1+nota2)/2);
6.      return 0;
7.  }
```

```
A sua média é: 9
```

A função *printf*() – caracteres e *strings*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      printf ("%s %s %s \n","teste1", "teste2", "teste3");
5.      printf ("A letra %c pronuncia-se %s \n", 'j', "jota");
6.      printf ("A letra \'%c\' pronuncia-se \'%s\'", 'j', "jota");
7.      return 0;
8.  }
```

```
teste1 teste2 teste3
A letra j pronuncia-se jota
A letra 'j' pronuncia-se 'jota'
```

A função *printf*() – *int*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 438;
5.      printf (" |%d|     \n", a);
6.      printf (" |%2d|    \n", a);
7.      printf (" |%6d|    \n", a);
8.      printf (" |%-6d|   \n", a);
9.      printf (" |%06d| \n", a);
10.     return 0;
11. }
```

```
|438|
|438|
|   438|
|438   |
|000438|
```

A função *printf*() – *float*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      float b = 65.1238;
5.      printf ("|%f|       \n", b);
6.      printf ("|%5f|      \n", b);
7.      printf ("|%9.1f|    \n", b);
8.      printf ("|%-9.1f|   \n", b);
9.      printf ("|%09.3f|  \n", b);
10.     return 0;
11. }
```

```
|65.123802|
|65.123802|
|     65.1|
|65.1     |
|00065.124|
```

Padrão de 6 casas decimais.

A função *printf*() – Tabela ASCII

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      printf ("%c %d %x %o \n", 'A' , 'A' , 'A' , 'A');
5.      printf ("%c %c %c %c  \n", 'A' , 65  , 0x41 , 0101);
6.      return 0;
7.  }
```

- Próprio Caracter
- Código ASCII em Decimal (base 10)
- Hexadecimal (base 16)
- Octal (base 8)

```
A 65 41 101
A A A A
```

A função *printf*() – Função

```
1.  #include <stdio.h>
2.  int Atualiza(int a)
3.  {
4.      a = a * 2 + 3;
5.      return a;
6.  }
7.  int main()
8.  {
9.      int x = 4;
10.     printf (" x atual = %d", Atualiza(x));
11.     return 0;
12. }
```

```
x atual = 11
```

A função *putchar*()

- **Função para impressão de caracteres isolados na saída padrão (*stdout* - tela).**
- **Definida na biblioteca padrão de C (stdio.h).**

Sintaxe: putchar (<caracter ou código ASCII do caracter em base decimal>);

A função *putchar*()

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char c = 'A';
5.      putchar(c);
6.      putchar('A');
7.      putchar(65);
8.      return 0;
9.  }
```

A função *scanf*()

- **Função para leitura de dados da entrada padrão (*stdin* - teclado), armazenado-os em variáveis.**
- **Definida na biblioteca padrão de C (stdio.h).**
- **Utiliza <u>códigos de formatação</u> semelhantes aos da função *printf*().**

<u>Sintaxe</u>: scanf("<string de entrada>", &<variável1>, &<variável2>, ...);

- **string de entrada = contém códigos de formatação que representam os tipos das variáveis. Pode apresentar caracteres que devem ser digitados junto com as variáveis.**
- **& (*address-of*) = endereço da posição de memória associada à variável.**
- **variável = variável que receberá o valor digitado. Para mais de uma variável, separar cada valor digitado por um espaço em branco, uma tabulação, um *enter*, ou pelo caracter definido em string de entrada.**

A função *scanf*() – Variável *char*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char c;
5.      printf ("Digite um caractere: ");
6.      scanf ("%c", &c);
7.      printf ("Caractere \'%c\' salvo no endereço %p",c, &c);
8.      return 0;
9.  }
```

```
Digite um caractere: A
Caractere 'A' salvo no endereço 0x7ffd0e028dff
```

A função *scanf*() – Variável *int*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int n;
5.      printf ("Digite um número: ");
6.      scanf ("%d", &n);
7.      printf ("Número %d salvo no endereço %p",n, &n);
8.      return 0;
9.  }
```

```
Digite um número: 6
Número 6 salvo no endereço 0x7ffe747e425c
```

A função *scanf*() – Variável *double*

```
1.  #include <stdio.h>
2.  #define PI 3.14
3.  int main()
4.  {
5.      double raio, comprimento;
6.      printf ("Quanto mede o raio? ");
7.      scanf ("%lf", &raio);
8.      comprimento = 2*PI*raio;
9.      printf ("O comprimento da circunferência é %5.2lf", comprimento);
10.     return 0;
11. }
```

```
Quanto mede o raio? 4
O comprimento da circunferência é 25.12
```

A função *scanf*() – Mais de uma variável

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      float nota1, nota2, nota3;
5.      printf ("Digite as três notas do aluno: ");
6.      scanf ("%f %f %f", &nota1, &nota2, &nota3);
7.      printf ("A média aritmética é %5.2f", (nota1 + nota2 + nota3)/3);
8.      return 0;
9.  }
```

```
Digite as três notas do aluno: 6 8 10
A média aritmética é  8.00
```

Separar os valores digitados com: espaço, *tab* ou *enter*

A função *scanf*()

Mais de uma variável com separador definido

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      float nota1, nota2, nota3;
5.      printf ("Digite as três notas do aluno: ");
6.      scanf ("%f , %f , %f", &nota1, &nota2, &nota3);
7.      printf ("A média aritmética é %5.2f", (nota1 + nota2 + nota3)/3);
8.      return 0;
9.  }
```

```
Digite as três notas do aluno: 2, 4, 6
A média aritmética é  4.00
```

Aqui os valores digitados devem ser separados por “,”

A função *scanf*()

Mais de uma variável com separador definido

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      float nota1, nota2, nota3;
5.      printf ("Digite as três notas do aluno: ");
6.      scanf ("%f , %f , %f", &nota1, &nota2, &nota3);
7.      printf ("A média aritmética é %5.2f", (nota1 + nota2 + nota3)/3);
8.      return 0;
9.  }
```

```
Digite as três notas do aluno: 2 4 6
A média aritmética é  0.67
```

Erro! Separador entre os valores digitados não respeitado.

A função *getchar*()

- **Função para leitura de caracteres isolados na entrada padrão (*stdin* - teclado).**
- **Definida na biblioteca padrão de C (stdio.h).**

Sintaxe: <variável de entrada> = getchar();

A função *getchar*()

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char c;
5.      printf ("Digite um caractere: ");
6.      c = getchar();
7.      printf ("Caractere \'%c\' salvo no endereço %p",c, &c);
8.      return 0;
9.  }
```

```
Digite um caractere: A
Caractere 'A' salvo no endereço 0x7ffd0e028dff
```

Operadores

Operadores são elementos funcionais que atuam sobre operandos e produzem um determinado resultado.

Unários → atuam sobre um único operando. Ex.: operador de incremento ++a;

Binários → atuam sobre dois operandos. Ex.: operador de soma a + b.

Tipos de operadores

- **Atribuição**
- **Aritméticos**
- **Aritmético de atribuição**
- **Incremento / decremento**
- **Relacionais**
- **Lógicos**

a + b

operandos

operador

Operador de Atribuição

- **O sinal de igual "=" é o operador de atribuição do C.**
- **Ele atribui o valor da direita à variável da esquerda.**
- **Este operador retorna o valor atribuído.**

Sintaxe: <variável> = valor;

Exemplos:

```
a = 2;          // a variável "a" recebe o valor 2
x = y = z = 10; // as variáveis "z", "y" e "x" recebem o valor 10
```

Operador de Atribuição

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 2;
5.      int x, y, z;
6.      x = y = z = 10;
7.      printf ("a = %d \n", a);
8.      printf ("x = %d y = %d z = %d \n", x, y, z);
9.      return 0;
10. }
```

```
a = 2
x = 10 y = 10 z = 10
```

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 2;
5.      int x = y = z = 10;
6.      printf ("a = %d \n", a);
7.      printf ("x = %d y = %d z = %d \n", x, y, z);
8.      return 0;
9.  }
```

```
main.c: In function ‘main’:
main.c:5:15: error: ‘y’ undeclared (first use in this function)
      int x = y = z = 10;
              ^
```

Operadores Aritméticos

Operador Binário	Efeito
+	Soma de dois números
-	Diferença entre dois números
*	Produto de dois números
/	Quociente de dois números
%	Resto da divisão de dois números

Podem ser usados com constantes e variáveis!

Quociente: fornece resultado inteiro apenas quando ambos os operandos são inteiros.

- **Ex.:** 7 / 2 = 3 e 7.0 / 2 = 3.5.

Resto: somente pode ser utilizado com operandos inteiros.

- **Ex.:** 7 % 2 = 1 e 7.0 % 2 = erro de compilação.

Operadores Aritméticos - *constantes* e *variáveis*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 7, b = 2;
5.      printf ("Soma: %d\n", a + 5);
6.      printf ("Subtração: %d\n", 5 - b);
7.      printf ("Multiplicação: %d\n", 5 * 10);
8.      printf ("Divisão: %d\n", a / b);
9.      printf ("Resto da divisão: %d\n", a % b);
10.     return 0;
11. }
```

```
Soma: 12
Subtração: 3
Multiplicação: 50
Divisão: 3
Resto da divisão: 1
```

Operadores Aritméticos de Atribuição

Operador Binário	Efeito
+=	a += 2; → a = a + 2;
-=	a -= 2; → a = a – 2;
*=	a *= 2; → a = a * 2;
/=	a /= 2; → a = a / 2;
%=	a %= 2; → a = a % 2;

Podem ser usados com variável à esquerda e constante, variável ou expressão à direita!

Instrução:

- <operando1> < operador aritmético > = <operando2>

Corresponde a:

- <operando1> = <operando1> < operador aritmético> < operando2>

Operadores Aritméticos de Atribuição

constante, variável e expressão

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 1, b = 2;
5.      printf ("%d \n", a += 3);       // a = a +3
6.      printf ("%d \n", a += b);       // a = a + b
7.      printf ("%d \n", a += b + 3);   // a = a + b + 3
8.      return 0;
9.  }
```

```
4
6
11
```

Operadores Aritméticos de Atribuição

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      float saldo = 100, aporte;
5.      printf ("Digite o valor a ser aportado: ");
6.      scanf ("%f", &aporte);
7.      printf ("Saldo = %6.2f", saldo += aporte);
8.      return 0;
9.  }
```

```
Digite o valor a ser aportado: 100
Saldo = 200.00
```

Operadores de Incremento / Decremento

Operador Unário	Efeito
++	Incremento
--	Decremento

Só podem ser usados com variável!

A diferença entre a pré e a pós-fixação aparece somente em expressões!

Pré-fixado: o símbolo vem antes do operando (ex.: ++a).

- **Neste caso a variável *a* é incrementada antes do seu valor ser usado.**

Pós-fixado: o símbolo vem depois do operando (ex.: a++).

- **Neste caso a variável *a* é incrementada depois do seu valor ter sido usado.**

Incremento / Decremento – *char*, *int*, *float*

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      char  c = 'a';
5.      int    x = 2;
6.      float y = 6.8;
7.      ++c;
8.      ++x;
9.      ++y;
10.     printf (" c = %c \n x = %d \n y = %4.2f \n", c, x, y);
11.     return 0;
12. }
```

```
c = b
x = 3
y = 7.80
```

Os resultados com o operador de decremento (--) são análogos!

Incremento / Decremento – pré e pós-fixado

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 4, b = 4;
5.      ++a;
6.      b--;
7.      printf ("a = %d, b = %d \n", a, b);
8.      return 0;
9.  }
```

```
a = 5, b = 3
```

Incremento / Decremento – pré e pós-fixado

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 4, b = 4;
5.      ++a;
6.      b--;
7.      printf ("a = %d, b = %d \n", a, b);
8.
9.      a = 4, b = 4;
10.     a++;
11.     --b;
12.     printf ("a = %d, b = %d \n", a, b);
13.     return 0;
14. }
```

```
a = 5, b = 3
a = 5, b = 3
```

Incremento / Decremento – pré e pós-fixado

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.          int a, b;
5.          a = 4;
6.          b = ++a;  // pré-fixado
7.          printf ("a = %d\n", a);
8.          printf ("b = %d\n\n", b);
9.          a = 4;
10.         b = a++;  // pós-fixado
11.         printf ("a = %d\n", a);
12.         printf ("b = %d\n", b);
13.         return 0;
14. }
```

```
a = 5
b = 5

a = 5
b = 4
```

b = ++a; ⇨

```
a = a + 1;
b = a;
```

b = a++; ⇨

```
b = a ;
a = a + 1;
```

Os resultados com o operador de decremento (--) são análogos!

Operadores Relacionais

Operador	Significado
==	Igual a
!=	Diferente de
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a

Fazem comparações!

Podem ser usados com constantes, variáveis e expressões!

Retornam um valor inteiro com significado lógico	**1 = verdadeiro** ➢ Ex.: ('A' != 'a') → 1 **0 = falso** ➢ Ex.: ('A'!= 'A') → 0

Operadores Relacionais

constantes, variáveis e expressões

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 1;
5.      char b = '2';
6.      printf ("%d \n", 'A' == 'a');
7.      printf ("%d \n", 'A' == 'A');
8.      printf ("%d \n", b == 2);
9.      printf ("%d \n", b == '2');
10.     printf ("%d \n", b == a + 1);
11.     return 0;
12. }
```

```
0
1
0
1
0
```

Cuidado com a diferença entre '2' e 2

Operadores Relacionais – Cuidado!

atribuição, comparação e negação

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int a = 2;
5.      printf ("Resultado = %d, %d, %d", a=2, a==2, a!=2);
6.      return 0;
7.  }
```

```
Resultado = 2, 1, 0
```

O comando de atribuição (a = 2) tem como retorno o valor atribuído!

Operadores Lógicos

Operador	Significado
&& (binário)	e
|| (binário)	ou
! (unário)	não

Usados com operandos lógicos (1 e 0) resultados, por exemplo, de uma comparação!

Tabela Verdade					
A	B	A && B	A || B	!A	!B
V	V	V	V	F	F
F	V	F	V	V	F
V	F	F	V	F	V
F	F	F	F	V	V

Operadores Lógicos

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      printf ("%d \n", ('A' == 'a') && ('A' == 'A'));    // F e V
5.      printf ("%d \n", ('A' == 'a') || ('A' == 'A'));    // F ou V
6.      printf ("%d \n", ! ('A' == 'a'));                  // não F
7.      return 0;
8.  }
```

```
0
1
1
```

Precedência de Operadores em Expressões

maior

menor

Operador	Tipo dos Operadores
! - ++ --	lógico (não), menos unário, incremento, decremento
* / %	aritméticos
+ -	
> >= < <=	relacionais
== !=	
&&	lógicos
||	
= += -= *= /= %=	aritméticos de atribuição

- Operadores de maior prioridade devem ser avaliados primeiro.
- Em caso de empate, a avaliação se faz da esquerda para a direita.
- O uso de parênteses em sub-expressões força a avaliação das mesmas com maior prioridade.

Precedência de Operadores em Expressões

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf ("%d\n",   'A' == 'A' || 'A' == 'a'   && 'A' == 'a'  );
5.     printf ("%d\n", ( 'A' == 'A' || 'A' == 'a' ) && 'A' == 'a' );
6.     return 0;
7. }
```

```
1
0
```

Precedência de Operadores em Expressões

```
1. #include <stdio.h>
2. int main()
3. {
4.     printf ("%d\n",  'A' == 'A' || 'A' == 'a'  && 'A' == 'a' ); // V ou F e F
5.     printf ("%d\n", ('A' == 'A' || 'A' == 'a') && 'A' == 'a' ); // (V ou F) e F
6.     return 0;
7. }
```

Operador lógico && tem precedência maior que ||.

```
1
0
```

Atividades

1. **Utilizando as funções *printf()* e *scanf()*, escreva um código para calcular:**

   a) **O equivalente em Fahrenheit (F) para uma temperatura dada na escala Celsius (C). Fórmula de conversão: F = 9/5 * C + 32. A temperatura em Celsius deve ser solicitada ao usuário. O equivalente em Fahrenheit deve ser apresentado na tela da seguinte forma: Celsius = ? equivale a Fahrenheit = ?.**

   b) **O valor de y como função de x, segundo a função y(x) = 3x + 1, num domínio real. O valor de x deve ser solicitado ao usuário. O valor de y deve ser apresentado na tela da seguinte forma: O valor de 'y' para 'x' igual a ? é: ?.**

   c) **A média aritmética de quatro números inteiros. Solicite os quatro números ao usuário com apenas duas linhas de comando. A média deve ser apresentada na tela da seguinte forma: Média aritmética entre ?, ?, ? e ? = ?.**

2. **Utilizando o compilador de sua preferência (ex: onlinegdb), implemente e teste os códigos apresentados nos slides 38, 41 e 46. Apresente pelo menos uma alteração em cada código em função da sua curiosidade.**

Referências

- **FORBELLONE, A. L. V., EBERSPACHER, H. F. Lógica de Programação – A Construção de Algoritmos e Estruturas de Dados, 3ª Edição, São Paulo, Pearson Prentice Hall, 2005.**
- **SOUZA, M. A. F., GOMES, M. M., SOARES, M. V., CONCILIO, R. Algoritmos e Lógica de Programação, 3ª Edição, São Paulo, Cengage, 2019.**
- **PUGA, S., RISSETTI, G. Lógica de Programação e Estrutura de Dados, 3ª. Edição, Prentice Hall, 2016.**
- **DEITEL, H. M., DEITEL, P. J. C: Como Programar. LTC, 2011.**
- **ASCENCIO, A. F. G., CAMPOS, E. A. V. Fundamentos da Programação de Computadores – Algoritmos, Pascal e C/C++. Pearson Prentice Hall, 2012.**
- **VAREJÃO, F. M. Introdução à Programação: Uma nova abordagem usando C. Campus, 2015.**
- **CELES, W., CERQUEIRA, R., RANGEL, J. L. Introdução a Estrutura de dados com Técnicas de Programação em C. Campus, 2016.**
- **MIZRAHI, V. V. Treinamento em Linguagem C – Curso Completo, 2ª Edição, Pearson Makron Books, 2008.**
- **SCHILDT, H., C Completo e Total, 3ª Edição, Makron Books, 1997.**