Teste de mesa e bibliotecas

Prof. Lilian Berton

São José dos Campos, 2019

Teste de mesa

- Verifica se o programa leva a um resultado esperado através da simulação de valores.
- Simula a execução de um algoritmo sem usar o computador, apenas papel e caneta.
- Importante para verificar a lógica do programa e identificar possíveis erros.

Passo-a-passo

- **Passo 1:** Primeira leitura para aproximação ao problema
- **Passo 2:** Segunda leitura
  - Anotações: identificar e anotar qual é o problema
  - Objetivos: identificar os objetivos. Ex: somar dois números
- **Passo 3:** Terceira leitura para revisão dos objetivos e conclusão
- **Passo 4:** Identificar quais são as variáveis necessárias
- **Passo 5:** Marcar os valores iniciais
- **Passo 6:** Escrever uma ideia base
- **Passo 7:** Testar a ideia do *passo 6* com alguns valores
- **Passo 8:** Identificar possíveis falhas na ideia e corrigi-las.

Teste das variáveis

- Identifique as variáveis envolvidas no programa.

- Crie uma tabela com linhas e colunas, tal que:
  - Cada coluna representará uma variável.
  - Cada linha corresponderá aos valores obtidos pelas variáveis.

a	b	nome	idade
1	0	Ana	20

Teste de mesa - exemplo 1

- Escreva um algoritmo para ler dois números e apresentar o resultado das 4 operações (adição, subtração, multiplicação e divisão).

```
#include <stdio.h>
main(void) {
    float a, b, ad, sub, mult, div;
    scanf("%f %f",&a,&b);
    ad = a+b;
    sub= a-b;
    mult = a*b;
    div = a/b;
    printf("O valor da subtracao é %f, da adicao é %f, da multiplicacao é %f e da divisao é %f", sub, ad, mult, div);
}
```

a	b	ad	sub	mult	div
12	3				
		15	9	36	4

Teste de mesa - exemplo 1

- Escreva um algoritmo para ler dois números e apresentar o resultado das 4 operações (adição, subtração, multiplicação e divisão).

```
#include <stdio.h>
main(void) {
    float a, b, ad, sub, mult, div;
    scanf("%f %f",&a,&b);
    ad = a+b;
    sub= a-b;
    mult = a*b;
    div = a/b;
    printf("O valor da subtracao é %f, da adicao é %f, da multiplicacao é %f e da divisao é %f", sub, ad, mult, div);
}
```

a	b	ad	sub	mul	div
2	3				
		5	-1	6	0.66

Teste de mesa - exemplo 2

- Escreva um algoritmo para ler dois números e trocar seus valores.

```
#include <stdio.h>
main(void) {
    int a, b;
    scanf (“%d %d”, &a,&b);
    a = b;
    b = a;
    printf(“Os valores de a e b são %d, %d”, a,b);
}
```

a	b
3	12
12	12

- Este programa exibe um resultado incorreto!

Teste de mesa - exemplo 2

- Escreva um algoritmo para ler dois números e trocar seus valores.

```
#include <stdio.h>
main(void) {
    int a, b, aux;
    scanf (“%d %d”, &a,&b);
    aux = a;
    a = b;
    b = aux;
    printf(“Os valores de a e b são %d, %d”, a,b);
}
```

a	b	aux
3	12	
12	3	3

- Este programa exibe um resultado correto!

Exemplo operador &&/and

- Escreva um algoritmo para ler dois números e apresentar o resultado da divisão se ambos forem positivos.

```
#include <stdio.h>
main(void) {
    int a, b;
    scanf (“%d %d”, &a,&b);
    if((a > 0) && (b > 0))
    printf(“A divisao de a por b: %d”, a/b);
}
```

a	b	a/b
0	0	
0	1	
-1	0	
10	5	

Exemplo operador &&/and

- Escreva um algoritmo para ler dois números e apresentar o resultado da divisão se ambos forem positivos.

```
#include <stdio.h>
main(void) {
    int a, b;
    scanf (“%d %d”, &a,&b);
    if((a > 0) && (b > 0))
    printf(“A divisao de a por b: %d”, a/b);
}
```

a	b	a/b
0	0	
0	1	
-1	0	
10	5	2

Exemplo operador II/or

- Escreva um algoritmo para ler dois números e apresentar o resultado da multiplicação se um deles for positivo.

```
#include <stdio.h>
main(void) {
    int a, b;
    scanf ("%d %d", &a,&b);
    if((a > 0) II (b > 0))
    printf("A multiplicacao de a por b: %d", a*b);
}
```

a	b	a*b
0	0	
0	1	
1	-1	
-1	-1	

Exemplo operador II/or

- Escreva um algoritmo para ler dois números e apresentar o resultado da multiplicação se um deles for positivo.

```
#include <stdio.h>
main(void) {
    int a, b;
    scanf ("%d %d", &a,&b);
    if((a > 0) II (b > 0))
    printf("A multiplicacao de a por b: %d", a*b);
}
```

a	b	a*b
0	0	
0	1	0
1	-1	-1
-1	-1	

Exemplo divisão inteiros e reais

```c
#include <stdio.h>

int main () {
  int i=4;
  int j=5;
  int k;
  float f = 5.0;
  float g;

  k = 6*(j/i); /* variável inteira k recebe resultado de expressão inteira */
  g = 6*(f/i); /* variável real g recebe resultado de expressão real */
  printf("1: k=%d g=%f\n", k, g);

  g = 6*(j/i); /* variável real g recebe resultado de expressão inteira */
  k = 6*(f/i); /* variável inteira k recebe resultado de expressão real */
  printf("2: k=%d g=%f\n", k, g);

  return 0;
```

i	j	k	f	g
4	5		5.0	

Exemplo divisão inteiros e reais

```c
#include <stdio.h>

int main () {
  int i=4;
  int j=5;
  int k;
  float f = 5.0;
  float g;

  k = 6*(j/i); /* variável inteira k recebe resultado de expressão inteira */
  g = 6*(f/i); /* variável real g recebe resultado de expressão real */
  printf("1: k=%d g=%f\n", k, g);

  g = 6*(j/i); /* variável real g recebe resultado de expressão inteira */
  k = 6*(f/i); /* variável inteira k recebe resultado de expressão real */
  printf("2: k=%d g=%f\n", k, g);

  return 0;
```

i	j	k	f	g
4	5		5.0	
		6		7.5
		7		6.0

Exemplo laço de repetição

- Escreva um algoritmo para calcular o fatorial de um número.

```
#include <stdio.h>

int main()
{
     int N, cont=1, fat=1;
     scanf("%d",&N);

     while (cont <= N) {
              fat = fat * cont;
              cont++;
     }
     printf("%d",fat);
   return 0;
}
```

N	cont	fat
4		

Exemplo laço de repetição

- Escreva um algoritmo para calcular o fatorial de um número.

```
#include <stdio.h>

int main()
{
     int N, cont=1, fat=1;
     scanf("%d",&N);

     while (cont <= N) {
            fat = fat * cont;
            cont++;
     }
     printf("%d",fat);
   return 0;
}
```

N	cont	fat
4	1	1
	2	1
	3	2
	4	6
	5	24

Indicador de passagem

```
# include <stdio.h>

int main () {
  int pos, i, x;

  pos = 0;
  i = 0;
  while (i<10) {
    printf ("Entre com x: ");
    scanf ("%d", &x);
    if (x > 0) {
      pos = 1;
    }
    i = i + 1;
  }
  if (pos == 0)
    printf ("Todos elems menores ou iguais a zero\n");
  else
    printf ("Pelo menos um elem. maior do que zero\n");

  return 0;
}
```

- **Pos é um indicador de passagem.**
- No final da repetição while, testa-se o valor de pos.
- Se o valor de pos é zero, então a condição x>0 nunca foi satisfeita, indicando que todos os elementos da sequencia são menores ou iguais a zero.
- Agora, se o valor de pos é um, em algum momento a condição x>0 foi satisfeita, indicando que pelo menos um elemento da sequencia é maior do que zero.

Indicador de passagem

```
# include <stdio.h>

int main () {
  int pos, i, x;

  pos = 0;
  i = 0;
  while (i<10) {
    printf ("Entre com x: ");
    scanf ("%d", &x);
    if (x > 0) {
      pos = 1;
    }
    i = i + 1;
  }
  if (pos == 0)
    printf ("Todos elems menores ou iguais a zero\n");
  else
    printf ("Pelo menos um elem. maior do que zero\n");

  return 0;
}
```

pos	i	x
0	0	-1
	1	-2
	2	-10
	3	-5
	4	-1
	5	-3
	6	-4
	7	5
	8	-6
	9	-7

Saída = ???

Uso de constantes

```
# include <stdio.h>

# define TRUE 1
# define FALSE 0

int main () {
  int pos, i, x;

  pos = FALSE;
  i = 0;
  while (i<10) {
    printf ("Entre com x: ");
    scanf ("%d", &x);
    if (x > 0) {
      pos = TRUE;
    }
    i = i + 1;
  }
  if (pos == FALSE)
    printf ("Todos elems menores ou iguais a zero\n");
  else
    printf ("Pelo menos um elem. maior do que zero\n");

  return 0;
}
```

Constantes!

Constantes são usadas para armazenar valores que **NÃO podem ser modificadas** durante a execução de um programa.

Uma constante precisa ser declarada, e para tanto usamos a diretiva de pré-processador **#define**.

Bibliotecas em C

- **Cabeçalho** (*header*) - tem extensão *.h* e possui a definição de funções, macros, variáveis e/ou constantes. Este tipo de arquivo usa a linguagem C e é essencial para a compilação dos programas que usam a biblioteca.

Cabeçalhos	Descrição
assert.h	Utilizado nas depurações de programas
complex.h	Define várias funções para manipular números complexos
ctype.h	Utilizado para testar e converter caracteres
errno.h	Utilizado na identificação e no tratamento de erros
fenv.h	Define funções e macros para manipulação de ponto flutuante
float.h	Especifica as características do ponto flutuante no sistema
inttypes.h	Define funções e macros para conversão entre tipos inteiros
limits.h	Especifica as características dos tipos de dados (byte, char, int)
locale.h	Define informações sobre localização (país, língua, mensagens, moeda, etc)
math.h	Define várias funções matemáticas

Bibliotecas em C

setjmp.h	Permite definir "non-local jumps" (interrompe a sequência normal, executa outras tarefas e retorna ao fluxo normal)
signal.h	Permite definir tratamento de sinais
stdarg.h	Permite acessar os argumentos de uma função quando o número de argumentos é desconhecido
stdbool.h	Define o tipo de dado booleano
stddef.h	Apresenta definições de tipos padrão (muitas dessas definições são também encontradas em outros cabeçalhos)
stdint.h	Define o tipos de dados inteiros
stdio.h	Permite realizar operações de entrada/saída
stdlib.h	Define várias funções de propósito geral como gerenciamento de memória dinâmica, geração de número aleatório, comunicação com o ambiente, aritmética de inteiros, busca, ordenação e conversão
string.h	Define funções para manipulação de strings
tgmath.h	Define macros para uso em diversas operações matemáticas
time.h	Define funções para ler e converter datas e horas

Vamos usar

Biblioteca *math*

- Fornece um conjunto de funções para operações matemáticas, tais como funções trigonométricas, hiperbólicas, logaritmos, potência e arredondamentos.
- **Todas as funções da biblioteca math.h retornam um valor do tipo double.**

Função	**Descrição do comando**
floor( )	arredonda para baixo
ceil( )	arredonda para cima
sqrt( )	calcula raiz quadrada
pow(variável, expoente)	potenciação
sin( )	seno
cos( )	cosseno
tan( )	tangente
log( )	logaritmo natural
log10( )	logaritmo base 10

Exercícios

1. Mostre com **o teste de mesa** o resultado das seguintes operações:

```
int i,j;
float y;

i = 5 / 3;
j = 5 / 3;

y = i / 2; /* divisão inteira (i e 2 são inteiros) */
y = i / 2.0; /* divisão em ponto flutuante (denominador real) */
y = i / j; /* divisão inteira (i e j são inteiros) */
y = (1.0 * i) / j; /* divisão em ponto flutuante (numerador real) */
y = 1.0 * (i / j); /* divisão inteira (i e j são inteiros) */
i = y / 2; /* parte inteira da divisão em i (divisão real, mas i é inteiro) */
```

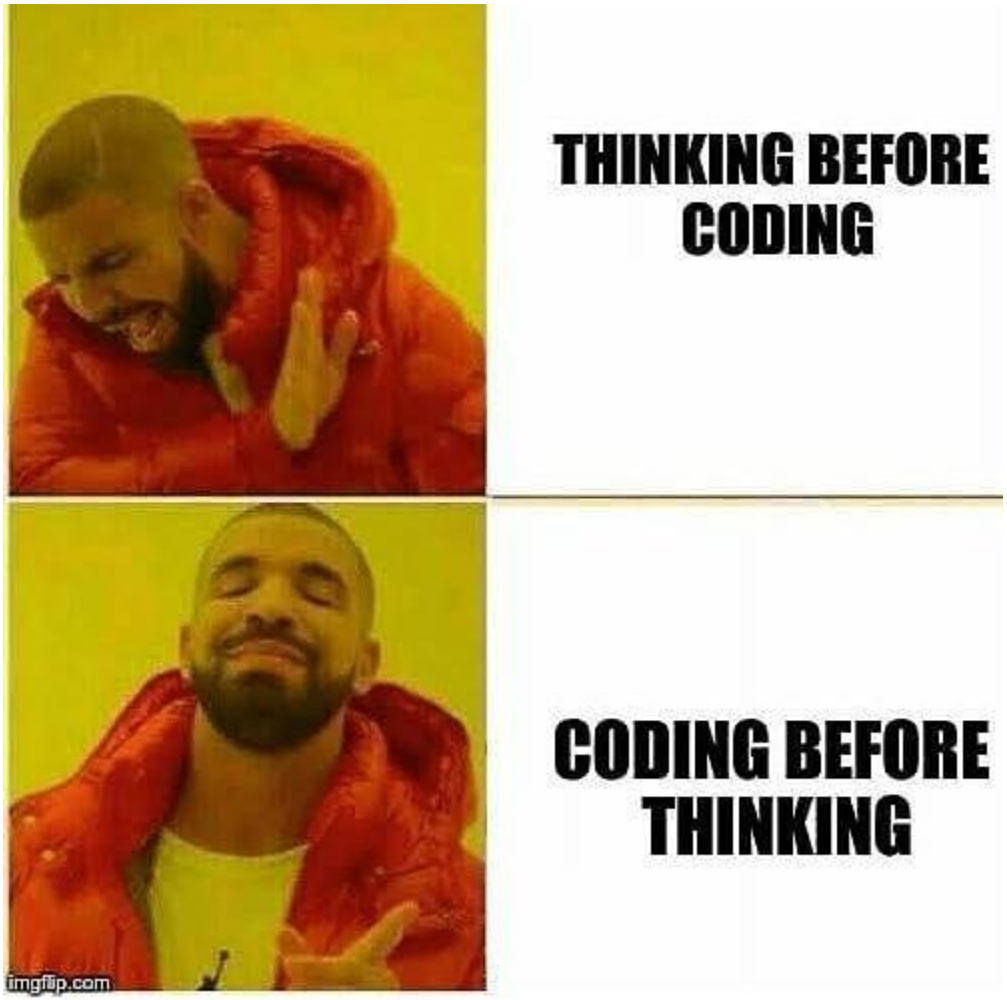
Exercícios

2. Faça um **programa e o teste de mesa** para resolver o seguinte problema:

   Dados um número inteiro n > 0 e uma sequência com n números inteiros, **determinar a soma dos inteiros positivos da sequência**. Por exemplo, para n=6 e para a sequência com n=6 números inteiros 6 −27 0 −5 84 -33 o seu programa deve escrever o número 90.

3. Faça um **programa e o teste de mesa** para resolver o seguinte problema:

   Dados um número inteiro n > 0 e uma sequência com n números inteiros, **determinar o menor número da sequência**. Por exemplo, para n=6 e para a sequência com n=6 números inteiros 6 −27 0 −5 84 -33 o seu programa deve escrever o número -33.


THINKING BEFORE CODING
CODING BEFORE THINKING
imgflip.com