UNISANTOS
Universidade Católica de Santos
Centro de Ciências Exatas, Arquitetura e Engenharia

Professor:	Ciro Cirne Trindade
Disciplina:	Introdução à Computação-II
Cursos:	Ciência da Computação/Sistemas de Informação

1ª Lista de Exercícios – Funções

1. Encontre o erro em cada um dos seguintes segmentos de programa e explique como ele pode ser corrigido:

   a)
   ```
   int quadrado(int x);
   {
      return x * x;
   }
   ```

   b)
   ```
   int soma(int x, y) {
     int resultado;
     resultado = x + y;
     return resultado;
   }
   ```

   c)
   ```
   void foo(float a) {
     float a;
     printf("%f\n", a);
   }
   ```

2. Qual a saída do programa abaixo?

```
#include <stdio.h>
void func1(void);
void func2(void);
int main() {
   printf("Inicio da main()\n");
   func1();
   printf("De volta a main()\n");
   func2();
   printf("Final da main()\n");
   return 0;
}

void func1() {
   printf("Em func1()\n");
}

void func2() {
   printf("Em func2()\n");
   func1();
}
```

3. Criar um programa que dados 3 números inteiros, utilize uma função do tipo `void` que receba esses números como parâmetro e imprima o maior deles.

4. Escreva uma função de protótipo `void retangulo(int a, int c);` que desenha no vídeo um retângulo formado por asteriscos (*) com *a* linhas de altura e *c* colunas de comprimento. Por exemplo, se for feita a seguinte chamada a função: `retangulo(5, 10);`

   A função deve desenhar no vídeo o seguinte retângulo:

```
**********
**********
**********
**********
**********
```

5. Escreva uma função de protótipo `void triangulo(int n, char ch);` que desenha no vídeo um triângulo invertido formado por *n* caracteres *ch* em sua base. Por exemplo, se for feita seguinte chamada a função: `triangulo(7, '#');`

   A função deve desenhar no vídeo o seguinte triângulo:

```
#######
 #####
  ###
   #
```

6. Escreva uma função para calcular e devolver o fatorial de um número natural passado como parâmetro.

7. Considere a função do exercício anterior e escreva um programa que solicita dois números naturais (*n* e *k*) ao usuário e calcula e imprime:

   a) O número de permutação P_n: $P_n = n!$

   b) O número de arranjos $A_{n,k}$: $A_{n,k} = \frac{n!}{(n-k)!}$

   c) O número de combinações $C_{n,k}$: $C_{n,k} = \frac{n!}{k!*(n-k)!}$

8. Escreva uma função de protótipo `double hipotenusa(double x, double y);` que calcula e devolve o comprimento da hipotenusa de um triângulo retângulo cujos catetos são dados pelos parâmetros *x* e *y*. Lembre-se que $hipotenusa = \sqrt{x^2 + y^2}$. Dica: utilize a função `sqrt` para obter a raiz quadrada.

9. A função `floor`, definida no arquivo *math.h*, arredonda seu argumento (um número do tipo `double`) para o maior inteiro que não seja maior que esse argumento, na prática, isso significa devolver a parte inteira do argumento. Entretanto, o valor de retorno da função `floor` é um `double`. Crie uma função de protótipo `int arredondarParaInt(double n);` que arredonda seu parâmetro *n* para o inteiro mais próximo. Dica: some 0.5 a *n* e utilize a função `floor`. Escreva um programa que leia vários números e use a função `arredondarParaInt` para arredondar cada um desses números para o inteiro mais próximo.

10. Escreva uma função de protótipo `double arredondar(double n, int c);` que arredonda o valor de *n* para um número com precisão de *c* casas decimais. Por exemplo, `arredondar(5.78351,1)` devolve `5.8`, `arredondar(5.78351,2)` devolve `5.78`,

`arredondar(5.78351,3)` devolve `5.784`. Dica: utilize a função `arredondarParaInt` passando seu argumento multiplicado por 10^c, e depois divida o valor de retorno da função por 10^c.

11. Escreva uma função que recebe um inteiro *m* e devolve `true` (verdadeiro) se *m* é primo ou `false` (falso), caso contrário.

12. Escreva um programa que leia um número inteiro não-negativo *n* e imprima os *n* primeiros números primos. Utilize a função da questão anterior.

13. Qual a saída do programa abaixo?

```
#include <stdio.h>
int f(int);
int main() {
     int x = 5;
     printf("%d %d\n", f(x+2), f(f(x+2)));
     return 0;
}
int f(int x) {
     return x + 2;
}
```

14. Qual a saída do programa abaixo?

```
#include <stdio.h>
int confusao(int, int);

int main() {
     int x = 2, y = 5;
     y = confusao(y, x);
     x = confusao(y, x);
     printf("%d %d\n", x, y);
     return 0;
}

int confusao(int x, int y) {
     x = 2 * x + y;
     return x;
}
```

15. Um número *a* é dito *permutação* de um número *b* se os dígitos de *a* formam uma permutação dos dígitos de *b*. Exemplo: 5412434 é uma permutação de 4321445, mas não é uma permutação de 4312455. Obs.: Considere que o dígito 0 (zero) não aparece nos números.

    a) Faça uma função `contadigitos` que dados como parâmetros um inteiro *n* e um inteiro *d*, $0 < d \leq 9$, devolve quantas vezes o dígito *d* aparece em *n*.

    b) Usando a função do item anterior, faça um programa que lê dois inteiros positivos *a* e *b* e responda se *a* é permutação de *b*.