Primeira lista de exercícios v1.02
Linguagens de Programação e Aplicações – SSC0300
Prof: Daniel Rodrigo Ferraz Bonetti

1. Escreva um programa que imprima a ordem reversa de caracteres de uma string digitada pelo usuário. Ex:
Entrada: “Hello World.”
Saída: “.dlroW olleH”

2. Faça um programa que imprima um triângulo isóscele à direita de altura n, digitada pelo usuário. Ex:
Entrada: n = 3
Saída:
```
*
**
***
```

3. Faça um programa que imprima um triângulo de lado, com altura 2n-1 e com largura n. Ex:
Entrada: n = 4
Saída:
```
*
**
***
****
***
**
*
```

4. Faça um programa que imprima um triângulo de altura n e com 2n-1 de largura. Ex:
Entrada: n = 6
Saída:
```
     *
    ***
   *****
  *******
 *********
***********
```

5. Faça uma função que verifique se um inteiro é positivo; a declaração da função deve ser: int isPositive(int i), e deverá retornar 0 se for negativo e 1 caso contrário.

6. Implemente uma função que calcula x^y (não usar as funções prontas, você deve desenvolver a lógica para o cálculo da potência). Você deve utilizar a função da seguinte maneira:
```
float a = raiseToPower(2, 3); // retorna 8
float b = raiseToPower(9, 2); // retorna 81
float raiseToPower(float f, int power); // declaração da sua função
```

7. Implemente uma função que verifica se um número é primo. Essa função deverá retornar 1 se o número for primo ou 0, caso contrário. Mostrar na tela se o número é ou não primo. Ex:
Entrada: 11
Saída: O número 11 é um número primo

8. Faça uma função que determina quantos números são primos abaixo de um valor n digitado pelo usuário. Ex:
Entrada: 12
Saída: Existem 5 números primos menores que 12.

9. Faça uma função que gere um número aleatório entre 0 e 1.
Declaração da função: float sample01()
Chamada: float f = sample01(); // deve retornar um número aleatório entre 0 e 1

10. Seguindo a mesma linha de raciocínio do exercício anterior, faça uma função que gere números aleatórios entre os valores a e b digitados pelo usuário.
Declaração da função: float sampleMinMax (float a, float b)
Chamada: float f = sampleMinMax(-50, 50) // retorna um valor aleatório entre -50 e +50

11. Faça um programa que gere n valores aleatórios no intervalo a e b. Certifique-se que o valor de b seja sempre maior do que o valor de a, apresentando uma mensagem de erro caso contrário. Ex:
Entrada: n = 3; a = -50; b = 50;
Saída: 35.45, -1,75, -26,48 // uma saída possível

12. Modifique o programa do exercício anterior para que, ao invés de imprimir os valores na tela, aloque um vetor dinamicamente com tamanho n e preencha suas posições com números aleatórios gerados no intervalo a e b.
Declaração da função: float *sampleMinMax(float a, float b, int n)
Chamada: float *f = sampleMinMax(-50, 50, 3) // deve retornar o ponteiro de um float para um vetor de float's com três valores aleatórios

13. Faça uma função que imprima o conteúdo de um vetor alocado dinamicamente com tamanho n.
Declaração da função: void printVector(float *f, int n)
Chamada: printVector(f, n); // deve imprimir o conteúdo do vetor f de tamanho n

14. Utilize a função do Exercício 13 para imprimir os elementos do vetor gerado aleatoriamente pelo Exercício 12.

15. Faça um programa que produza a seguinte saída, dados dois valores x e y digitados pelo usuário. Ex:
Entrada: x = 10; y = 5
Saída:

x	y	expressões	resultados
10	5	x=y+3	8
10	5	x=y-2	3
10	5	x=y*5	25
10	5	x=x/y	2
10	5	x=x%y	0

16. Faça um programa que leia três notas e dê o conceito de acordo com as regras:
-se o conceito médio = 90% => conceito=A
-se o conceito médio >= 70% e <90% => conceito=B
-se o conceito médio >= 50% e <70% => conceito=C
-se o conceito médio < 50% => conceito=F

17. Faça um programa que imprima a seguinte saída a partir de um valor n digitado pelo usuário: Ex:
Entrada: n = 7
Saída:

```
       1******
       12*****
       123****
       1234***
       12345**
       123456*
       1234567
```

18. Faça um programa que solicite ao usuário digitar um valor inteiro positivo. O programa deverá termina quando um número negativo for digitado.

19. Implemente uma calculadora com o seguinte menu:

```
  MENU
          1. Adição
          2. Subtração
          3. Multiplicação
          4. Divisão
          5. Módulo
Escolha uma opção: 1
Entre com dois números: 12 15
Resultado: 27

Continuar? y
```

O programa também pergunta ao usuário se quer continuar a execução. Se o usuário digitar 'y' então o programa segue com a execução ou termina caso outro caractere seja digitado.

20. Implemente uma função que imprima os elementos de uma matriz alocada dinamicamente.
Declaração da função: void printMatrix(int **matrix, int rows, int columns)
Para alocar uma matriz dinamicamente, utilize os comandos:

```
int **m = (int**)calloc(sizeof(int*), rows); // aloca um ponteiro para as 'rows' linhas
for (int i=0; i<rows; i++)
    m[i] = (int*)calloc(sizeof(int), columns); // aloca um int para cada coluna e de cada linha da matriz
```

Chamada: printMatrix(m, rows, columns); // imprime a matriz

21. Faça um programa que crie uma matriz de tamanho n*n, em que o valor de n deve ser digitado pelo usuário, e preencha os elementos dessa matriz com a sequência de números crescentes de 1 até n*n. Ao final do programa, imprima a matriz (utilize a função do Exercício 20). Ex:
Entrada: n = 5
Saída:

```
1     2     3     4     5
6     7     8     9     10
11    12    13    14    15
16    17    18    19    20
21    22    23    24    25
```

22. Faça um programa que imprima uma matriz com dimensões n*n, em que n é digitado pelo usuário, e preencha os elementos da seguinte maneira. Ex:
Entrada: n = 5
Saída

```
 0     1     1     1    1
-1     0     1     1    1
-1    -1     0     1    1
-1    -1    -1     0    1
-1    -1    -1    -1    0
```

Dica: Para imprimir os valores alinhados, pode ser utilizado \t dentro do printf ou então "% d" (perceba que há um espaço entre o % e o d, para alinhar os números que possuem sinal e os que não possuem). Para mais informações sobre alinhamentos com o printf consulte: http://www.cplusplus.com/reference/cstdio/printf/

23. Faça uma função que retorne uma matriz de tamanho n*n em que a diagonal principal é igual a 1 e todos os outros elementos são iguais a 0. Ex:
Entrada: n = 4
Saída:
```
1     0     0     0
0     1     0     0
0     0     1     0
0     0     0     1
```
Dica: utilize a função que imprime matriz do Exercício 20.

24. Faça um programa que leia o nome de n países e mostre o país que tem o nome mais comprido digitado.
Dica: utilizar a função strlen da biblioteca string.h

25. Crie uma estrutura de dados capaz de armazenar a cor de um pixel dada em RGB (red, green e blue). A cor de um pixel em RGB é definida pela intensidade de vermelho, verde e azul. Tais valores podem variar de 0 a 255 (valores inteiros) em que valores mais baixos representam a ausência da cor e valores mais altos (próximos a 255) representam a presença da cor. Por exemplo, a cor branca é definida pelo RGB (255; 255; 255), enquanto que a cor preta é definida por (0; 0; 0). Ex:
Entrada: Entre com o RGB: 0 100 200
Saída: Vermelho = 0
```
       Verde    = 100
       Azul     = 200
```

26. Utilizando a estrutura do exercício anterior, implemente um programa que leia a cor de n pixels digitados pelo usuário e após a leitura dos valores RGB, mostre na tela os índices do pixel mais escuro e do mais claro bem como seus respectivos valores RGB.

27. O programa a seguir mostra o maior valor de n valores digitados pelo usuário. Complete os espaços em branco para que o programa funcione corretamente. Ex:
Entrada/Saída do programa:
```
  Quantidade de valores: 3
  Valor 1: 21
  Valor 2: 12
  Valor 3: 4
  O maior valor é 21.
```

```
#include<stdio.h>
#include<conio.h>

_____________________data, int n){
    int *max=data;
    int i;
    for(i=_; i<n; i++){
        if(_____________) *max=_______;
    }
    return ___;
}

int main() {
    int n,i,*p;
    printf("Quantidade de valores ");
    scanf("%d",&n);
    int *arr = _____________________________;

    for(i=0;i<n;i++) {
      printf("Valor %d:",i+1);
```

```
      scanf("%d", _______);
    }

    p = findMax(___,n);
    printf("O maior valor é %d", __);
    free(___);
    return 0;
}
```

28. Escreva um programa que faça adição de duas frações e mostre o resultado em forma de fração. O programa deverá solicitar que o usuário entre com a Fração 1 e a Fração 2. O numerador e o denominador devem ser separados por espaço. Utilize uma struct para receber os elementos da fração. Ex:

```
Entrada:
  Fração 1 (numerador denominador): 1 2
  Fração 2 (numerador denominador): 2 5
Saída:
  Resultado: 9/10
```

29. Faça um programa que sorteie um valor entre 1 e 100. Depois, peça que o usuário tente adivinhar esse número sugerindo se o número digitado pelo usuário é maior ou menor que número sorteado. O programa terminará depois que o usuário acertar o número. Deverá exibir na tela também o número de tentativas necessárias que o usuário precisou para acertar o número. A estrutura básica do programa pode ser:

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
      // suas variáveis aqui
      int num;
      srand(time(NULL)); // semente aleatória
      num = (rand()%100)+1; // obtém um número aleatório
      // escreva a rotina aqui
      return 0;
}
```

30. Escreva um programa que aloque um vetor dinamicamente de tamanho n digitado pelo usuário. Preencha esse vetor com valores da potencia de 2, indo de 1 até n. Por fim, faça uma função que imprima (Exercício 13) o conteúdo desse vetor alocado dinamicamente. Utilize a função pow(x,y) da biblioteca math.h. Ex:

```
Entrada: n = 6
Saída:
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
```

31. Faça uma função que crie uma sequência (chamada seq) de números dados os limites inferior e superior bem como a quantidade de subdivisões desse vetor. Ex:

```
Entrada: a = 1.0; b = 1.5; n = 10
Saída: 1.00 1.06 1.11 1.17 1.22 1.28 1.33 1.39 1.44 1.50
```

32. Faça um programa que, dado um certo vetor x de valores reais, calcule f(x), em que f(x) = seno(x). Ex:

```
float *x  = seq(0, 10, 20)      // função criada no exercício anterior
float *fx = computeFX(x, 20);   // função que retorna o ponteiro com os senos calc.
printVector(fx);                // Exercício 13
```

33. Crie uma estrutura de dados conforme a seguir:

```
typedef struct {
    float *seno;
    float *cosseno;
} Funcs;

typedef struct {
    float *x;
    int    n;   // número de pontos avaliados
    Funcs  fx;
} Points
```

Depois utilize a mesma ideia do exercício anterior para calcular o seno e o cosseno e armazenar na estrutura Points. Por fim, faça uma função que imprima na tela os valores de x e todos os fx's calculados. A declaração da função deve ser:
`void printPoints(Points p)`

Aviso: (opcional) Você pode criar apenas um projeto em sua IDE preferida (Visual Studio, Codeblocks, Dev etc) e, dentro da função main(), criar uma função para cada exercício. Ex:

```
void exer01() {
    // conteúdo do Exercício 01 aqui
}

void exer02() {
    // conteúdo do Exercício 02 aqui
}

void exer03() {
    // conteúdo do Exercício 03 aqui
}

.
.
.

void exerN() {
    // conteúdo do Exercício N aqui
}

int main() {

    // exer01(); // Você pode comentar os exercícios que já rodou
    // exer02();
    // exer03();
    ...();
    exerN();
}
```

Boa sorte ;)