Ficha de Trabalho N.º 2 (versão 2024/25)

Objetivos: Estudo e Aplicação das Estruturas de Repetição (ciclos).

Proposta de Resolução

1 - Conceba e escreva o algoritmo de um programa que leia um número inteiro e calcule todos os seus múltiplos inferiores a 100. Implemente o algoritmo em linguagem C.

Pseudodódigo (para a solução 1)

```
Entradas: num;
Saídas: Mensagem com os múltiplos de num < 100.
Variáveis: num, mult: REAL;
      INICIO
          FAZER
               ESCREVER('Introduza um número inteiro positivo: ');
               LER (num);
           ENQUANTO (num <=0)
           mult <- num;
           ENQUANTO (mult < 100) FAZER
               ESCREVER('O múltiplo é: ', mult);
               mult <- mult + num;
           FIM-ENQUANTO
      FIM.
```

Programa

```
// Exercício 1) Conceba e escreva o algoritmo de um programa que leia um número inteiro
// e calcule todos os seus múltiplos inferiores a 100.
// Implemente o algoritmo em linguagem C.
 #include<stdio.h>
 #include<locale.h>
 int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int num, mult;
    do
    {
       printf("Introduza número inteiro positivo: ");
       scanf("%d", &num);
    }while(num<=0);
    printf("Solução 1.....\n");
    mult = num;
    while (mult < 100)
    {
       printf("%d\n", mult);
       mult = mult + num;
    }
    printf("--------\n\n");
          printf("Solução 2.....\n");
    mult = num;
    if (num < 100)
    {
       do
       {
          printf("%d\n", mult);
          mult = mult + num;
       } while (mult < 100);
    }
    return 0;
```

```
}
```

2 - Elabore um programa que determine todos os números pares entre dois números inteiros *ni* e *nf* (*ni<nf*).

Pseudodódigo (para a solução 1)

Entradas: ni, nf;
Saídas: Mensagem com os números pares entre ni e nf;
Variáveis: ni, nf, aux, par, i: INTEIRO;

```
INICIO
    ESCREVER('Introduza dois números inteiros, ni<nf: ');
    LER (ni, nf);
    SE (ni > nf) ENTÃO
        aux <- nf;
        nf <- ni;
        ni <- aux;
    FIM-SE
    ESCREVER('Os números pares entre ', ni, ' e ', nf, ' são:');
    SE (ni RESTO_DIV_ZERO 2 = 0) ENTÃO
        Par <- ni;
    SENÃO
        par <- ni + 1;
    FIM-SE
    PARA i DESDE par ATÉ nf SALTO 2 FAZ
        ESCREVER ('o número par é: ', i);
    FIM-PARA
FIM.
```

Programa

```
// Exercício 2 - Elabore um programa que determine todos os números pares
// entre dois números inteiros ni e nf (ni<nf).
#include<stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int ni, nf, par,aux,i;
    printf("Introduza dois números inteiros, ni<nf: ");
    scanf("%d%d", &ni, &nf);
    //colocar o menor em ni
    if(ni>nf) //há que trocar
    {
       aux=nf;
       nf=ni;
       ni=aux;
    }
    printf("Os pares entre %d e %d são: \n",ni,nf);
    if (ni % 2 == 0)   //verifica se é divisível por 2, ou seja, se é par
       par = ni;
    else
       par = ni + 1;  //ni é ímpar, o primeiro par í o número seguinte
    for(i=par; i<=nf; i=i+2)  //pode escrevr-se como i+=2
    {
       printf("%d\n", i);
    }
}
```

3 - Elabore um algoritmo que peça ao utilizador para introduzir um número entre 0 e 9 e, enquanto não for introduzido um valor válido, a leitura deve ser repetida.

Pseudodódigo (para a versão 1)

Entradas: n;
Saídas: Mensagem mostrando o número inserido;
Variáveis: `n: INTEIRO;`

```
INICIO
    FAZER
        ESCREVER('Introduza um número entre 0 e 9: ');
        LER (n);
```

```
        ENQUANTO (n < 0 || n > 9)
        ESCREVER('O número especificado foi: ', n);
FIM.
```

Programa

```
// Exercício 3 - Versão 1 - Utiliza do ... while (condição) -
// Elabore um programa que peça ao utilizador para introduzir um número entre 0 e 9 e,
// enquanto não for introduzido um valor válido, seja repetida a leitura.

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int n;
   do
   {
      printf("\nIntroduza um número entre 0 e 9: ");
      scanf("%d",&n);
   }

 while (n<0 || n>9);
   printf ("\nO número especificado foi %d\n\n", n);
   return 0;
}

// Exercício 3 - Versão 2 - Utiliza do while (condição).
// Elabore um programa que peça ao utilizador para introduzir um número entre 0 e 9 e,
// enquanto não for introduzido um valor válido, seja repetida a leitura.

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
 setlocale(LC_ALL, "Portuguese");    int n=-1;
   // ou
   // printf("\nIntroduza um número entre 0 e 9: ");
   // scanf(" %d",&n);
   while (n<0 || n>9)
   {
      printf("\nIntroduza um numero entre 0 e 9: ");
      scanf(" %d",&n);
   }
   printf ("\nO n˙mero especificado foi %d\n\n", n);
   return 0;
}
```

4 - Conceba um algoritmo e crie o programa respectivo, que leia uma sequência de números inteiros positivos e determine quantos números são pares e quantos são ímpares. A finalização da sequência de números é indicada, introduzindo-se um número negativo.

Pseudocódigo (para a versão 1)

Entradas: n;
Saídas: contPar, contImpar;
Variáveis: n, conrPar, contImpar: INTEIRO;

```
INICIO
   FAZER
         ESCREVER('Introduza um número inteiro, positivo ou nulo (para finalizar,
                  deverá inserir um número negativo: ');
         LER (n);
         SE (n > 0) ENTÃO
               SE (n Resto_Divisao_Inteira 2) = 0 ENTÃO
                     contPar ← contPar + 1;
               SENÃO
```

```
                        contImpar ← contImpar + 1;
                FIM-SE
            ENQUANTO ( n >= 0)
            ESCREVER('Foram introduzidos ´, contPar, ` números pares e `,
                        contImpar, ` números ímpares.);
        FIM.
```

Programa

```
// Exercício 4 - Verão 1 - Utiliza do... while (condição).
// Elabore um algoritmo para um programa que leia uma sequência de números inteiros
// positivos e determine quantos números são pares e quantos são ímpares.
// A finalização da sequência de números é indicada introduzindo-se um número negativo.
#include <stdio.h>
#include<locale.h>
#include<stdlib.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   system("clear"); //limpa o ecrã, se o programa em Mac for executado,
                // clicando no executável gerado na compilação
   int n, contPar, contImpar, i=1;
   contImpar = 0;
   contPar = 0;
   printf("    *** Cálculo do número de valores pares e ímpares, incluídos numa sequência
              de números positivos inseridos ***\n");
   printf("                        ---> Para parar, deverá introduzir um
                  número negativo <---\n\n");
   do
   {
      printf("\t\tIntroduza o %d.º número da sequência: ", i);
      scanf("%d", &n);
      if (n > 0){   // se o utilizador inserir 0 (embora não devesse, pois o enunciado
                  // indica que deverá introduzir números pares ou negativos),
                  // o 0 seria aceite e considerado número par.
         if(n % 2 == 0)
            contPar++;   //contPar = contPar+1 ou ainda, contPar+=1;
         else
            contImpar++;
      }
   } while (n >= 0);
   printf("\n\t\tForam introduzidos %d números pares e %d números ímpares.\n",
                                                                  contPar, contImpar);
   return 0;
}

// Exercício 4 - Versão 2 - Utiliza while (condição).
// Elabore um algoritmo para um programa que leia uma sequência de números
// inteiros positivos e determine quantos números são pares e quantos são ímpares.
// A finalização da sequência de números é indicada introduzindo-se um número negativo.
#include <stdio.h>
#include<locale.h>
#include<stdlib.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   system("clear"); //limpa o ecrã, se o programa em Mac for corrido, clicando no
                        // executável gerado na compilação
   int n=-1, contPar=0, contImpar=0, i=1;
   printf("    *** Cálculo do número de valores pares e ímpares,
```

```
                    incluídos numa sequência de números positivos inseridos ***\n");
    printf("                ---> Para parar, deverá introduzir um número
                                                                    negativo <---\n\n");
    printf("\t\tIntroduza o %d.º número da sequência: ", i);
    scanf("%d", &n);
    while (n>=0)
    {
        if (n > 0){    // se o utilizador inserir 0 (embora não devesse, pois o enunciado
                       // indica que deverá introduzir números positivos ou negativos),
                       // o 0 seria aceite e considerado número par.
            if(n % 2 == 0)
                contPar++;   //contPar = contPar+1 ou ainda, contPar+=1;
            else
                contImpar++;
            i++;
        }
        printf("\t\tIntroduza o %d.º número da sequência: ", i);
        scanf("%d", &n);
    }
    printf("\t\tForam introduzidos %d número(s) par(es) e %d número(s) ímpar(es).\n\n",
                                                                    contPar, contImpar);
}
```

5 - Desenvolva um programa que determine os *n* primeiros múltiplos de um número inteiro *m*.

Programa

```
// Exercício 5
// Elabore um programa que determine os n primeiros múltiplos de um número inteiro m.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int n, m, mult,i;
    printf("Valor base (m): ");
    scanf("%d", &m);
    printf("Quantos múltiplos? ");
    scanf("%d", &n);
    mult = m;
    for(i=1; i<=n; i++)
    {
        printf("O %dº múltiplo de %d é: %d\n", i, m, mult);
        mult = mult + m;
    }
    return 0;
}
```

6 - Conceba e elabore o algoritmo e crie o respetivo programa que calcule a soma dos N primeiros números inteiros positivos, escrevendo, em cada iteração, o total acumulado.

Pseudodódigo

Entradas: n;
Saídas: soma;
Variáveis: `n, soma, i: INTEIRO;`

```
INICIO

    ESCREVER('Introduza um número inteiro positivo: ');
    LER (n);
    FAZER
```

```
            ESCREVER('Insira um número inteiro positivo: ');
            LER (n);
        ENQUANTO (n <= 1);
        SOMA ← 0;
        PARA i DESDE 1 ATÉ n FAZER
            soma ← soma +i;
            ESCREVER ('A soma na interação ', i, 2 é: ', soma);
        FIM-PARA
        ESCREVER('O valor final da soma dos números inteiros até ', n, ' é: ', soma);
FIM.
```

Programa

```
// Exercício 6
// Elabore o algoritmo e crie o respetivo programa que calcule a soma dos N primeiros
// números inteiros positivos, escrevendo, em cada iteração, o total acumulado.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
        setlocale(LC_ALL, "Portuguese");
        int n, soma,i;
        do
        {
        printf("Insira um número inteiro positivo: ");
        scanf("%d", &n);
        } while (n <= 1);
        soma = 0;
        for (i = 1; i <= n; i++)
        {
        soma = soma + i;
        printf("A soma na iteração %d é: %d\n", i, soma);
        }
        printf("\nO valor final da soma dos números inteiros até %d é: %d\n\n", n, soma);
        return 0;

}
```

7- Escreva um programa em C que calcule a tabuada de um número inteiro n dado pelo utilizador. A tabuada deve aparecer no monitor no formato:

n x _1 = _n*1
n x _2 = _ n*2
...
n x 10 = n*10

Obs.: Atenção à formatação da 2.ª e 3.ª colunas (devem estar alinhadas à direita).

Programa

```
// Exercício 7
// Escreva um programa em C que calcule a tabuada de um número inteiro i dado pelo
// utilizador. A tabuada deve aparecer no monitor no formato:
// i x  1 =  i
// i x  2 = 2i
// ...
// i  10 = 10i

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int i, n;
   printf("\n Quer saber a tabuada de que número? ");
   scanf("%d", &n);
   for (i=1; i <= 10; ++i)
      printf("\t%d x %2d = %2d\n", n, i, n*i);
   printf("\n");
```

```
    return 0;
}
```

8- Modifique o programa da alínea anterior de modo a calcular a tabuada de todos os números de 2 a 10, fazendo uma pausa depois de escrever cada uma delas.

Programa

```
// Exercício 8
// Modifique o programa da alínea anterior de modo a calcular a tabuada de todos
// os números de 2 a 10, fazendo uma pausa depois de escrever cada uma delas.
// Mais uma vez, a tabuada deve aparecer no monitor no formato:
// i x 1 =  i
// i x 2 = 2i
// ...
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   char c;
   int i, n;
   for (n=2; n<=10; ++n)
   {
      printf("------------------\n");
      printf("\tTabuada do %d\n", n);
      for (i=1;i<=10;++i)
         printf("\t%d x %2d = %2d\n", n, i, n*i);
      printf("\n Digite ENTER para continuar ");
      scanf("%c", &c);                    // espaço antes de %c para obrigar a parar,
                                          // mesmo que tenha um enter no buffer
   }
   printf("\n------------------\n");
   return 0;
}
```

9- Escreva um programa em C que leia um número inteiro positivo N e calcule o maior número par P tal que a soma de todos os números pares até P seja inferior a N. Por exemplo, se for dado o valor 57 para N, então o resultado será P = 14, pois 2 + 4 + 6 + 8 + 10 + 12 + 14 = 56 < 57 e 2 + 4 + 6 + 8 + 10 + 12 + 14 + 16 >= 57.

Programas:

```
// Exercício 9 - Versão 1 (do while (condição) - Escreva um programa em C que leia
// um número inteiro positivo N e calcule o maior número par P tal que a soma de
// todos os números pares inferiores a P seja inferior a N.
// Por exemplo, se for dado o valor 57 para N, então
// o resultado será P=14, pois 2 + 4 + 6 + 8 + 10 + 12 + 14 = 56.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int numero, par, soma;
   soma = 0;
   par = 0;
   printf("Por favor, introduza um numero inteiro positivo: ");
   scanf("%d", &numero);
   while (soma < numero)
   {
      par = par + 2;
      soma = soma + par;
   }  //o último número par é o que faz ultrapassar o número introduzido e,
```

```
        // consequentemente, termina o ciclo.
    printf ("O maior número par é: %d", par - 2);  // o valor correcto será o
                    // valor  do par anterior, pois o actual ultrapassou numero
    return 0;
}

// Exercício 9 - Versão 2 (com while... condição)
// Escreva um programa em C que leia um número inteiro positivo N e calcule o maior número
// par P, tal que a soma de todos os números pares inferiores a P, seja a inferior a N.
// Por exemplo, se for dado o valor 57 para N, então
// o resultado será P=14, pois 2 + 4 + 6 + 8 + 10 + 12 + 14 = 56.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int numero, par, soma;
    soma = 0;
    par = 0;
    printf("Por favor introduza um número inteiro positivo: ");
    scanf("%d", &numero);
    do
    {
        par = par + 2;
        soma = soma + par;
    }
    while (soma < numero);
    printf ("O maior número par: %d", par - 2);
    return 0;
}

// Exercício 9 - Versão 3 (com for - ainda que não o mais adequado e formal)
// - Escreva um programa em C que leia um número inteiro positivo N
// e calcule o maior número par P tal que a soma de todos os números pares inferiores a P
// seja inferior a N. Por exemplo, se for dado o valor 57 para N, então
// o resultado será P=14, pois 2 + 4 + 6 + 8 + 10 + 12 + 14 = 56.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int numero=1, par, soma;
    soma = 0;
    par = 0;
    do{
        printf("\nPor favor introduza um numero inteiro positivo: ");
        scanf("%d", &numero);
    }while(numero < 0);

    for(par=0; soma < numero; par+=2)
    {
        soma = soma + par;
    }
    printf ("\nO maior número par é : %d\n", par - 4);
    return 0;
}
```

10- a) Elabore um algoritmo e implemente-o para calcular o somatório $\sum_{i=1}^{n} i!$

Obs. Atenção que não deve especificar um valor para n superior a 12.

Pseudodódigo (para a solução do método da força bruta)

Entradas: n;
Saídas: soma;
Variáveis: n, soma, i, j, fact: INTEIRO;

```
INICIO
    ESCREVER('("*** CÁLCULO DO SOMATÓRIO DE I=1 ATÉ N DE I FACTORIAL, ***');
    ESCREVER('      *** UTILIZANDO O MÉTODO DA "FORÇA BRUTA" ***');
    ESCREVER('  ');
    SOMA ← 0;
    FAZER
         ESCREVER('Qual o valor de N?: ');
         LER (n);
     ENQUANTO (n <= 1);

    PARA i DESDE 1 ATÉ n FAZER
         Fact ← 1;
         PARA j DESDE 1 ATÉ i FAZER
              fact ← fact * i;
         FIM-PARA
         soma ← soma + fact;
         ESCREVER('O factorial de ', i, ' = 'fact');
         ESCREVER('O valor do somatório quando i = ', i, ' é: ', soma);
     FIM-PARA
     ESCREVER('Na versão "método da força bruta", o somatório de i=1 até ', n,
                ' de i factorial = ', soma);
FIM.
```

Programa (utilizando o método da "força bruta")

```
// Exercício 10
// Elabore um algoritmo e implemente-o para calcular o somatório ∑ de 1=1 até N de i!.
// Esta solução utiliza o denominado método da "força bruta", ou seja, não faz qualquer esforço
// em tentar reaproveitar a computação do valor do factorial do número anterior, para calcular o seguinte.
// Atenção que não devem especificar um valor para n superior a 12. Experimentem para 13, 14, 15 e
// observem o que acontece ao valor do somatório e do factorial, em cada iteração e tentem perceber porquê.
// Tentem perceber porquê (dica: lembrem-se da representação em binário dum int e quais
// os valores mínimos e máximos admitidos).

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int soma=0, i=1, n=1, fact=1, j=0;
    printf("*** Cálculo do somatório de i=1 até N de i factorial, ***\n");
   printf("      *** utilizando o método da 'força bruta' ***\n\n");
   do{
      printf("\nQual o valor de N (N>=1)? ");
      scanf("%d", &n);
   } while (n<=1);
   for(i=1;i<=n;i++)
   {
      //para cada i, calcular i!
      fact=1;
      for(j=1;j<=i;j++)
      {
         fact=fact*j;
      }
```

```
        soma=soma+fact;
        printf("Factorial de %d = [%d]\n", i, fact);
        printf("O valor do somatório quando i = %d é: [%d]\n\n", i, soma);
    }
    printf("Na versão 'método da força bruta', o somatorio de i=1 até %d
                        de i factorial = %d\n\n", n, soma);
}
```

b) Experimente o programa para n=13. Observa algo de estranho em algum valor apresentado? O quê?

R.: Experimentando para n=13 o somatório vai resultar num valor negativo! Estranho!

c) Agora indique n=14 ou 15 e observem o que acontece ao valor do somatório e do factorial, em cada iteração. Para todas estas situações, procure perceber porquê.

Para n=14, já ocorre o erro no valor do factorial e no valor do somatório.

Dica: Lembrem-se da representação em binário dum int e quais os valores mínimos e máximos admitidos.

d) Procurem uma solução e alterem o programa, experimentando agora com os valores de n que não resultavam em valores correctos.

Para resolver o problema, para valores de n um pouco maiores, pode usar-se o tipo long, para fact e soma.

A nova solução apresenta-se, seguidamente.

Nova versão, com fact e soma tipo long.

```
// Exercício 10 - Versão 2, usando para fact e soma o tipo long.
// Elabore um algoritmo e implemente-o para calcular o somatório ∑ de 1=1 até N de i!.
// Esta solução utiliza o denominado método da "força bruta", ou seja, não faz qualquer esforço
// em tentar reaproveitar a computação do valor do factorial do número anterior, para calcular o seguinte.
// Atenção que não devem especificar um valor para n superior a 12. Experimentem para 13, 14, 15 e v
// observem o que acontece ao valor do somatório e do factorial, em cada iteração e tentem perceber porquê.
// Tentem perceber porquê (dica: lembrem-se da representação em binário dum int e quais
// os valores mínimos e máximos admitidos).

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int i, j, n;
    long soma=0, fact=1;
    printf("*** Cálculo do somatório de i=1 até N de i factorial, ***\n");
    printf("      *** utilizando o método da 'força bruta' ***\n\n");
    do{
        printf("\nQual o valor de N (N>=1)? ");
        scanf("%d", &n);
    } while (n<=1);
    for(i=1;i<=n;i++)
    {
        //para cada i, calcular i!
        fact=1;
        for(j=1;j<=i;j++)
        {
            fact=fact*j;
        }
        soma=soma+fact;
        printf("Factorial de %d = %ld\n", i, fact);
        printf("O valor do somatório quando i = %d é: %ld\n\n", i, soma);
    }
    printf("Na versão 'método da força bruta', o somatório de i=1 até %d de i factorial =
```

```
                                                                %ld\n\n", n, soma);
}
```

e) Será que a solução é fiável para valores muito mais elevados de n? Porquê?

Não! Como o valor do factorial cresce exponencialmente e, obviamente, soma também , a solução não permite valores de n muito maiores.

f) Será que haverá uma solução melhor? Qual e experimentem-na, para verificar se é o é mesmo.

Solução alternativa é usar um float ou melhor, um double, mas, aí, há o problema da precisão. Arredondando a 0 casas decimais, provavelmente conseguir-se-ão valores inteiros incorrectos.

Outra solução algorítmica, mais elegante e eficiente:

```
// Exercício 10
// Elabore um algoritmo e implemente-o para calcular o ∑ de i=1 até N de i!.
// São apresentadas agora 3 soluções, utilizando uma solução mais elegante para o cálculo
// do factorial de n.
// Esta solução utiliza a denominada "técnica de programação dinâmica" que consiste em
// utilizar um valor anterior de um cálculo, para conseguir computar o seguinte,
// evitando-se repetir todos os cálculos realizados até ao momento.
// Atenção que o problema da limitação do valor a especificar para n persiste,
// embora o cálculo seja muito mais rápido.
// Serão apresentadas 3 soluções, empregando todas as estruturas de repetição existentes.

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int soma=0, i=1, n=1, f=1;
   // solução a) versão com do ... while (condição)
   printf("*** Cálculo do somatório de i=1 até N de i factorial ***\n");
   printf("  *** utilizando a técnica de Programação Dinâmica ***\n\n");
   do{
      printf("\nQual o valor de N (N>=1)? ");
      scanf("%d", &n);
   } while (n<=1);
   i=1;
   do{
      printf("\nQual o valor de N (N>=1)? ");
      scanf("%d", &n);
   } while (n<=1);
   i=1;
   f=1;  // variável para guardar o valor do factorial calculado e a usar na iteração seguinte
         // inicia a 1 pois 1 é o elemento neutro da multiplicação
   do
   {
      f=f*i; // calculo do factorial de i, aproveitando o cálculo do factorial de i-1
      soma+=f; // somatório do factorial de i
      printf("O valor do factorial de %d = [%d]\n", i, f);
      printf("O valor do somatório quando i = %d é: [%d]\n\n", i, soma);
      i++;
   } while (i<=n);
   printf("Na versão a), o somatorio de i=1 até %d de i factorial = [%d]\n", n, soma);
   printf("\n");

   // Solucao b) versão com while (condição) ...
   i=1;
```

```
    f=1;
    soma=0;
    while (i<=n)
    {
        f=f*i; // calculo do factorial de i
        soma+=f; // somatorio do factorial de i
        printf("O valor do factorial de %d = [%d]\n", i, f);
        printf("O valor do somatório quando i = %d é: [%d]\n\n", i, soma);
        i++;
    }
    printf("Na versão b), o somatorio de i=1 até %d de i factorial = [%d]\n", n, soma);
    printf("\n");

    // Solucao c) versão com for(...)
    soma=0;
    f=1;
    for(i=1; i<=n; i++)
    {
        f = f * i;
        soma = soma + f;
        printf("O valor do factorial de %d = [%d]\n", i, f);
        printf("O valor do somatório quando i = %d é: [%d]\n\n", i, soma);
    }
    printf("Na versão c), o somatorio de i=1 até %d de i factorial = [%d]\n", n, soma);
    return 0;
}
```

11- a) Elabore um programa que calcule o valor do seguinte somatório: $\sum_{i=1}^{N} \frac{2^i}{i!}$.

Programa (solução a))

```
// Exercício 11 - solução 1)
// Elabore um programa que calcule o valor do seguinte somatório: ∑ de 1 até N de 2^i/i!.
// Solução já utilizando o método da programação dinâmica, no cálculo quer da
// potência de 2, quer do factorial de i.
// No entanto, o problema verificado no exercício anterior persiste:
// só funciona para valores de n baixos, dado que o factorial de n grande dá origem
// a um valor que não pode ser guardado num int.
// Pode-se usar um long, mas, mesmo assim, o problema persiste para valores maiores de n.
// Assim, há que encontrar um algoritmo mais elegante (transformando e simplificando a
// fórmula de cálculo).
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int i, pote, fact, N;
    float soma;
    printf("Qual o valor de N? ");
    scanf("%d", &N);
    soma = 0;
    pote=1;
    fact=1;
    for (i = 1; i <= N; i++)
    {
        pote = pote * 2;
        fact = fact * i;
        soma = soma + pote * 1.0 / fact;  //multiplicar por 1.0 para fazer a divisão real
```

```
        printf("O valor do factorial de %d = [%d]\n", i, fact);
        printf("O valor do somatório quando i = %d é: [%10.7f]\n\n", i, soma);
    }
    printf("\O valor do somatório de 1 até % de 2^i/i = %.7f\n", soma);
    return 0;
}
```

b) Haverá uma solução mais elegante que resolva, inclusivamente, o problema da limitação do valor de N? Procure-a, implemente-a e teste-a para os valores que não eram admissíveis, anteriormente e mesmo para valores muito maiores,

Programa (solução b)

Dedução da solução:
$2^i/i! = 2 \times 2^{i-1} / i \times (i-1)!$: se repararmos, $2^{i-1}/ (i-1)!$ é o termo anterior.
Então, o termo actual, $2^i/ (i)! = 2^{i-1}/ (i-1)!$ (o termo anterior) * 2/i.
Pode então usar-se esta expressão para o cálculo do termo actual, tornando a solução bastante mais elegante.

```
// Exercício 11 - solução 2
// Elabore um programa que calcule o valor do seguinte somatório: ∑ de 1 até N de 2^i/i!.
// Solução mais elegante, utilizando o método da programação dinâmica, mas a outro nível:
// Em vez de calcular o numerador e denominador separadamente, utilizando o método da
// programação dinâmica, vai aproveitar-se o cálculo completo da parcela anterior,
// bastando, depois, multiplicá-la por 2/i.
// Em termos de eficiência no cálculo, à priori, não é fácil perceber qual será mais
// rápida. Só experimentando com um valor de n elevado para as 2 soluções.
// Mas a grande vantagem é que se evita o problema do cálculo do factorial que levava aos
// erros que se viram atrás, quando N>12.

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int i, n;
    float soma, parc;
    printf("Qual o valor de N? ");
    scanf("%d", &n);
    soma = 0.0;
    parc = 1.0;
    for (i = 1; i <= n; i++)
    {
        parc = parc * 2*1.0 / i;
        soma = soma + parc;
        printf("O valor de parc quando i = %d é: [%10.7f]\n", i, parc);
        printf("O valor do somatório quando i = %d é: [%10.7f]\n\n", i, soma);
    }
    printf("\nA soma final é: %10.7f\n", soma);
}
```

```
// Exercício 11 - solução 3 - Usar parc como double
//              A partir de N=35, o valor do somatório não se altera, pois a precisão
//                      de parc deixa o valor em 0
// Elabore um programa que calcule o valor do seguinte somatório: ∑ de 1 até N de 2^i/i!.
// Solução mais elegante, utilizando o método da programação dinâmica, mas a outro nível:
// Em vez de calcular o numerador e denominador separadamente, utilizando o método da programação dinâmica,
// vai aproveitar-se o cálculo completo da parcela anterior, bastando, depois, multiplicá-la por 2/i.
// Senão, veja-se, como se chegou à nova solução:
// 2^i/i! = 2^(i-1) * 2 / (i-1)! * i : se repararmos, 2^(i-1)/(i-1)! é o termo anterior.
// Então, o termo actual, 2^i/(i)! = 2^(i-1)/(i-1)!(o termo anterior) * 2/i.
// Pode então usar-se esta expressão para o cálculo do termo actual, tornando a solução
// bastante mais elegante.
// Em termos de eficiência no cálculo, à priori não é fácil perceber qual será mais rápida. Só experimentando
// um valor de n elevado para as 2 soluções.
// Mas a grande vantagem é que se evita o problema do cálculo do factorial que leva aos erros
// que se viram atrás quando N>12.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int i, N;
    double soma,parc;
    printf("Qual o valor de N? ");
    scanf("%d", &N);
    soma = 0;
    parc = 1;
    for (i = 1; i <= N; i++)
    {
        parc = parc * 2*1.0 / i;
        soma = soma + parc;
        printf("O valor de parc quando i = %d é: [%.30f]\n", i, parc);
        printf("O valor do somatório quando i = %d é: [%.30f]\n\n", i, soma);
    }
    printf("\nO valor do somatório de 1 até %d de 2^i/i = %.30f\n", N, soma);
}
```

12) Escreva um programa em C que peça um número inteiro positivo e escreva todos os seus divisores, do maior para o menor.

Programa

```
// Exercício 12
// Escreva um programa em C que peça um número inteiro positivo e escreva
// todos os seus divisores, por ordem decrescente de valor.
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");int n, i;
    do
    {
        printf("Introduza um número (n>0): ");
        scanf("%d", &n);
    }while (n <= 0);
    printf("Os divisores de %d são: \n", n);
    for(i=n/2; i>=1; i--) // do maior para o menor
    {
        if(n % i == 0) // i é divisor
        {
            printf("%d, ",i);
        }
    }
    printf("\n");
    return 0;
}
```

13) Escreva um programa que calcule a soma da seguinte série de N termos:

$$1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{6} + \frac{1}{8} - \frac{1}{10} + \cdots$$

Programa

```
// Exercício 13
// Escreva um programa que calcule a soma da seguinte série de N termos:
// 1 - 1/2 + 2/4 - 3/8 + 4/16 - 5/32 + ....
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int n,i, sinal;
   float s;
   do{
      printf("\n Qual é o número de termos que pretende para a série (>=1)? ");
      scanf("%d",&n);
   } while (n <= 1);
   sinal=1;
   s=1;
   for (i=1;i<=n;++i)
   {
      sinal=-sinal;
      s=s+sinal/(2.0*i);
      // printf("\t  termo %d = %0.4f\n ",i,sinal/(2.0*i));
   }
   printf("\t\n soma = %0.4f\n ",s);
   printf("FIM\n");
   return 0;
}
```

14) Crie uma cópia do programa elaborado em resposta ao exercício anterior, e altera-a, para que calcule a soma da seguinte série de N termos:

$$1 - \frac{1}{2} + \frac{2}{4} - \frac{3}{8} + \frac{4}{16} - \frac{5}{32} + \cdots$$

Programa

```
// Exercício 14
// Crie uma cópia do programa elaborado em resposta ao exercício anterior, e altera-a,
// para que calcule a soma da seguinte série de N termos:
// 1 - 1/2 + 2/4 - 3/8 + 4/16 - 5/32 + ....
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int sinal, i, n, numer, den;
   float s;
   do{
      printf("\n Qual é o número de termos que pretende para a série (>=1)? ");
      scanf("%d",&n);
   } while (n <= 1);
   sinal=1;
   s=1;
   numer=0;
   den=1;
   for (i=1;i<=n;++i)
   {
      sinal=-sinal;
      numer+=1;
      den*=2;
      s=s+sinal*(numer*1.0/den);
```

```
        // printf("\t  termo %d = %0.4f\n ",i,sinal *(numer*1.0/den));
    }
    printf("\t\n soma = %0.4f\n ",s);
    printf("FIM\n");
    return 0;
}
```

15) Escreva um programa que coloque no ecrã meia árvore de Natal com asteriscos. O número de ramos deverá ser indicado pelo utilizador.

Exemplos com três e quatro ramos:

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

Programa

```
// Exercício 15 - versão 1
// Escreva um programa que coloque no ecrã meia árvore de Natal com asteriscos.
// O número de ramos deverá ser indicado pelo utilizador.
// Exemplos com três e quatro ramos:
//                                  *
//          *                       * *
//          * *                     * * *
//          * * *                   * * * *

#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int nRamos, i, j;
    do
    printf("Quantos ramos pretende para a árvore? ");
    scanf("%d", &nRamos);
    for (i = 1; i <= nRamos; i++) //para cada ramo i
    {
        printf("\t");
        // Vai desenhar-se a árvore, com um tab antes, para não aparecer
        // logo no início do ecrã.
        // Assim, vai-se inserir um \t e, depois, i símbolos
        for (j = 1; j <= i; j++) //escrever i *
        {
            printf("*"); // escreve 1 símbolo (*); também pode ser: printf("%c",'*');
        }
        printf("\n"); // muda de linha antes de passar a desenhar o próximo ramo
    }
    return 0;
}
```

Outra alternativa, uma alteração do programa para que os ramos fiquem com o seguinte aspecto:

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

```
// Exercício 15 - versão 2
// Escreva um programa que coloque no ecrã meia árvore de Natal com asteriscos.
// O número de ramos deverá ser indicado pelo utilizador.
```

```
// Exemplos com três e quatro ramos:
//                                  *
//              *                  * *
//             * *                * * *
//            * * *              * * * *
#include <stdio.h>
#include<locale.h>
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int nRamos, i, j;
    printf("Quantos ramos pretende para a árvore? ");
    scanf("%d", &nRamos);
    for (i = 1; i <= nRamos; i++) //para cada ramo i
    {
        // Vai desenhar-se a árvore, com um tab antes, para não aparecer
        // logo no início do ecrã.
        // Cada ramo (linha) tem n-i espaços e i símbolos
        printf("\t");

        for (j = 1; j <= nRamos - i; j++)//escrever n-i espaços
        {
            printf(" "); //escreve 1 espaço; alternativa: printf("%c",' ');
        }
        for (j = 1; j <= i; j++) //escrever i *
        {
            printf("*"); //escreve 1 símbolo (*); alternativa: printf("%c",'*');
        }
        printf("\n");// muda de linha, para passar a desenhar um novo ramo;
    }
    return 0;
}
```

16) Sendo dado o valor de N, compreendido entre 1 e 9, elabore e teste um programa que produza uma pirâmide de números de acordo com o exemplo seguinte para N=3.

```
    1
   121
  12321
```

Programa

Versão 1

```
// Exercício 16 - Criação da árvore na sua versão original.
// Sendo dado o valor de N, compreendido entre 1 e 9, elabore e teste um programa que
// produza uma pirâmide de números de acordo com o exemplo seguinte, para N=3.
//           1
//          121
//         12321
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 9
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int n, i, j;
    do
    {
```

```
        printf("Indique o numero de degraus da pirâmide, sem exceder %d: ",MAXL);
        scanf("%d",&n);
    } while (n>MAXL);
    for (i=1;i<=n;++i)  // para cada degrau i
    {
        printf("\t"); // escreve um tab para não desenhar a pirâmide encostada à margem esquerda do ecrã
        for (j=1;j<=MAXL-i;++j)  // repete MAXL - i vezes
            printf(" "); // escreve os espaços à esquerda até inciar o desenho da pirâmide
        for (j=1;j<=i;++j){
            printf("%d",j);  // escreve os números a subir
        }
        for (j=i-1;j>0;j--){
            printf("%d",j);  // escreve os números a descer
        }
        printf("\n");       // muda de linha, depois de a escrever, para desenhar o próximo degrau
    }
    return 0;
}
```

Versão 2

```
// Exercício 16 - Versão 2 - Criação da pirâmide, agora com um máximo de 19 degraus.
// Sendo dado o valor de N, compreendido entre 1 e 19, elabore e teste um programa que produza uma
// pirâmide de números de acordo com o exemplo seguinte, para N=3.
//            1
//           121
//          12321
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 19
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int n, i, j;
    do
    {
        printf("Indique o numero de linhas da piramide, sem exceder %d: ",MAXL);
        scanf("%d",&n);
    } while (n>MAXL);
    for (i=1;i<=n;++i)  // para cada degrau i
    {
        for (j=1;j<=MAXL-i;++j)  // repete MAXL - i vezes
            printf(" "); // escreve os espaços à esquerda até inciar o desenho da pirâmide
        for (j=1;j<=i;++j){
            if(j<=9)
                printf("%d",j);  // números a subir (enquanto o número a escrever <=9)
            else
                printf("%d",j/2); // a partir da linha 10, como já não há mais números,
                                  // no meio da árvore, repete números 5, 6, 7, 8 e 9.
        }
        for (j=i-1;j>0;j--){
            if(j<=9)
                printf("%d",j);  // números a descer (enquanto o número a escrever <=9)
            else
                printf("%d",j/2);
        }
        printf("\n"); // muda de linha, depois de a escrever, para desenhar o próximo degrau;
    }
```

```
    printf("\n\n"); // muda de linha, depois de a escrever, para desenhar o próximo degrau;
    return 0;
}
```

Versão 3

```
// Exercício 16 - Versão 3
// Sendo dado o valor de N, compreendido entre 1 e 19, elabore e teste um programa que produza
// uma pirâmide de números, mas podendo ser estreita (igual às anteriores) ou
// larga (com um espaço entre cada número),d e acordo com o exemplo seguinte, para N=3.
//           1        1
//          121      1 2 1
//         12321    1 2 3 2 1
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 19
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int n, i, j;
   char c;
   do
   {
      printf("Indique o numero de deraus da piramide, sem exceder %d: ",MAXL);
      scanf("%d",&n);
   } while (n>MAXL);
   printf("Pirâmide estreita<E> ou larga <L>:");
   scanf(" %c: ",&c);
   for (i=1;i<=n;++i)  // para cada linha i
   {
      switch (c)
      {
         case 'E':  // pirâmide estreita
            for (j=1;j<=MAXL-i;++j)  // repete MAXL - i vezes
               printf(" "); // escreve espaços à esquerda
            for (j=1;j<=i;++j){
               if(j<=9)
                  printf("%d",j);  // números a subir (enquanto o número a escrever <=9)
               else
                  printf("%d",j/2); // a partir da linha 10, como já não há mais números,
                              // no meio da pirâmide, repete números 5, 6, 7, 8 e 9.
            }
            for (j=i-1;j>0;j--){
               if(j<=9)
                  printf("%d",j);  // números a descer
               else
                  printf("%d",j/2);
            }
            break;
         case 'L':   // pirâmide larga
            for (j=1;j<=MAXL-i;++j)
               printf("  "); // escreve espaços à esquerda com o dobro dos espaços (2)
                                          // para ficar a pirâmide mais larga
            for (j=1;j<=i;++j)
               printf("%2d",j);  // números a subir, mas colocando o número com um espaço antes
                                          // para a pirâmide ficar mais larga
            for (j=i-1;j>0;j--)
               printf("%2d",j);  // números a descer, mas colocando o número com um espaço antes
                                          // para a pirâmide ficar mais larga
            break;
         default:
```

```
            printf("Tipo de pirâmide inválida!!!");
        }
        printf("\n"); // muda de linha, depois de a escrever, preparando o desenho do próximo degrau;
    }
}
```

17) Elabore um programa para efectuar o cálculo de a^b, sendo a um número real e b um número inteiro positivo:

a) Utilizando um ciclo while;

b) Utilizando um ciclo for.

Programa – Alínea a) Solução 1:

```
// Exercício 17 - a) Solução 1: com while(condição) ...
// Elaborar um programa para efectuar o cálculo de a elevado a b, sendo a um número real e b um número inteiro.
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 20
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int b, b1;
    float a, res;
    printf("\nIntroduza o valor da base: "); scanf("%f",&a);
    do{
        printf("\nIntroduza o valor do expoente (valor>0): ");
        scanf("%d",&b);
    } while (b<0);
    b1=b; // guardar o valor inicial de b, para depois apresentar na mensagem final
    res=1; // qualquer número levantado a 0 = 1
    while (b > 0){
        res *= a;
        b --;
    }
    printf("\n\n%0.3f elevado a %d = %0.3f\n\n", a, b1, res);
    return 0;
}
```

Programa – Alínea a) Solução 2:

```
// Exercício 17 - a) Solução 2: com do ... while(condição)
// Elaborar um programa para efectuar o cálculo de a elevado a b, sendo a um número real
// e b um número inteiro.
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 20
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int b, b1;
    float a, res;
    printf("\nIntroduza o valor da base: "); scanf("%f",&a);
    do{
```

```
        printf("\nIntroduza o valor do expoente (valor>0): ");
        scanf("%d",&b);
    } while (b<0);
    b1=b; // guardar o valor inicial de b, para depois apresentar na mensagem final
    res=1; // qualquer número levantado a 0 = 1
    do
    {
        res *= a;
        b --;
    } while (b > 0);
    printf("\n\n%0.3f elevado a %d = %0.3f\n\n", a, b1, res);
    return 0;
}
```

Programa – Alínea b)

```
// Exercício 17 - b) Solução com for(..,..,..)
// Elaborar um programa para efectuar o cálculo de a elevado a b, sendo a um número real e
// b um número inteiro.
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 20
int main(int argc, const char * argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int i, b;
    float a, res;
    printf("\nIntroduza o valor da base: "); scanf("%f",&a);
    do{
        printf("\nIntroduza o valor do expoente (valor>0): ");
        scanf("%d",&b);
    } while (b<=0);
    res=1; // qualquer número levantado a 0 = 1
    for(i=1; i<=b; i++)
        res *= a;
    printf("\n\n%0.3f elevado a %d = %0.3f\n\n", a, b, res);
    return 0;
}
```

18) Elabore um programa que calcule a média de um conjunto de valores reais positivos, considerando os seguintes casos:

a) O número de parcelas, *n*, deve ser previamente pedido ao utilizador.

b) O programa deve pedir continuamente números reais e parar quando for introduzido um número negativo, calculando depois a média dos valores introduzidos (exceto o negativo).

Programa

```
// Exercício 18.a) e b)
// Elabore um programa que calcule a média de um conjunto de valores reais positivos, considerando os seguintes casos:
// a) O número de parcelas, n, deve ser previamente pedido ao utilizador;
// b) O programa deve pedir continuamente números reais e parar quando for introduzido
//    um número negativo, calculando depois a média dos valores introduzidos (não considerando
//    aquele que terminou o processo de inserção).
#include <stdio.h>
#include<locale.h>
#define _CRT_SECURE_NO_WARNINGS
#define MAXL 20
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int opcao, n, i, nParcelas;
   float soma, valor;
   printf("\t\t*** Calculo da média de um conjunto de valores reais positivos ***\n");
   printf("Pretende:\n  1-> Especificar previamente o número de valores positivos de que pretende
                  calcular a média?\n");
   printf("  2-> Ir inserindo valores positivos, terminando com um número <=0?\n");
   printf("      --> Opção pretendida: ");
   scanf("%d", &opcao);
   printf("\n");
   switch (opcao)
   {
   case 1:
      do
      {
         printf("Quantos valores positivos? ");
         scanf("%d", &n);
      } while (n < 1);
      soma = 0; nParcelas = n;
      for (i = 1; i <= n; i++)
      {
         printf("Insira o %d.º valor: “, i);
         scanf("%f", &valor);
         if (valor > 0)
            soma = soma + valor;
         else//foi introduzido valor negativo ou nulo
            nParcelas--;//nParcelas=nParcelas-1; não se acrescenta nada à soma
                  //e o número parcelas diminui
      }
      if (nParcelas > 0)
         printf("\nA média dos %d valores inseridos = %.4f\n", n, soma / nParcelas);
      else //nParcelas=0
         printf("Não inseriu quaisquer valores válidos!\n");
      break;
   case 2:
      soma = 0.0;
      n = 0;
         printf("-> P. f., vá inserindo os sucessivos valores de que pretende calcular a
                  média,\n");
         printf("-> Para terminar, insira um valor <0\n\n");
      do
      {
         printf("Insira o %d.º valor: ", n+1);
         scanf("%f", &valor);
         if (valor > 0.0)
```

```
        {
            soma = soma + valor;
            n++;
        }
    } while (valor > 0.0);
    if (n > 0)
        printf("\nA média dos %d valores inseridos = %.4f\n", n, soma / n);
    else
        printf("Não foram inseridos quaisquer valores válidos!\n");
    break;
  default:
    printf("Opção inválida!!\n");
    break;
  }
  return 0;
}
```

19) Escreva um programa em C que implemente o jogo do palpite:

Um jogador escreve um número inteiro entre 0 e 100 (o programa deve obrigar a que seja um número dentro destes limites) e um segundo jogador tenta adivinhar esse número.
O programa deve indicar se o número dado no palpite é superior ou inferior ao número a adivinhar e quantas tentativas foram necessárias.

Programa

```
// Exercício 19
// Elaborar um programa em C que implemente o jogo do palpite:
// "Um jogador escreve um número inteiro entre 0 e 100 (o programa deve obrigar a que seja
// um número dentro destes limites)
// e um segundo jogador tenta adivinhar esse número".
// O programa deve indicar se o número dado no palpite é superior ou inferior ao número a adivinhar.

#include <stdio.h>
#include<locale.h>
#include<stdlib.h>
#define _CRT_SECURE_NO_WARNINGS
int main(int argc, const char * argv[])
{
  setlocale(LC_ALL, "Portuguese");
  int numero, palpite, tentativas;
  system("clear"); // Limpa o ecrã: funciona em MacOS, se o programa executável for chamado
                   // no terminal ou no finder, clicando no executável, na pasta build/debug
            // para windows, usar a mesma intrução ou system("cls");
  do
  {
    printf("\n\nJogador 1: Qual o número a encontrar [0 ... 100] ? ");
    scanf("%d", &numero);
  } while (numero<0 || numero >100); // ou } while (!(numero>=0 && numero <=100));
  // system("cls"); // limpa ecrã em Windows e DevC++
  system("clear"); //limpa o ecrã
  tentativas=0;
  do
  {
    tentativas++;
    do
    {
      printf("\n\nJogador 2: Qual o seu palpite [0 .. 100] ? ");
      scanf("%d",&palpite);
    } while (numero<0 || numero > 100);  // ou } while (!(numero>=0 && numero <=100));

    if (palpite > numero)
    printf("\nO seu palpite é MAIOR que o número a encontrar!");
    else if (palpite < numero)
        printf("\nO seu palpite é MENOR que o número a encontrar!");
     else printf("\n\n**** PARABÉNS!!!! ----- ACERTOU!****\n");
```

```
    } while (palpite!=numero);
    printf("Foram necessárias %d tentativas\n\n", tentativas);
    return 0;
}
```

20) Escreva um programa para calcular a data da Páscoa de um ano indicado pelo utilizador, permitindo-lhe repetir o cálculo enquanto o pretender.

Dica: Procurem o Algoritmo de Gauss para o cálculo do Dia da Páscoa.

Programa

```
// Exercício 20
// Conceba e desenvolva um programa para calcular a data da Páscoa de um ano indicado pelo utilizador,
// permitindo-lhe repetir o cálculo, enquanto ele pretender.
#include <stdio.h>
#include<locale.h>
#include<stdlib.h>
#define _CRT_SECURE_NO_WARNINGS
int main(int argc, const char * argv[])
{
   setlocale(LC_ALL, "Portuguese");
   int ano, mes, dia, a, b, c, d, e;
   char opcao;
   do
   {
      do
      {
         printf("Indique um ano, no intervalo [1900, 2099]: ");
         scanf("%d",&ano);
      } while (ano <=1900 || ano>2099);
      a= ano % 19;
      b= ano % 4;
      c= ano % 7;
      d= (19*a+24) % 30;
      e= (2*b+4*c+6*d+5) % 7;
      if (d+e>9)
      {
         mes=4;
         dia=d+e-9;
      }
      else
      {
         mes=3;
         dia=22+d+e;
      }
      printf("\n\n A Páscoa do ano %d ocorre em %d/%d \n\n",ano,dia,mes);
      printf("Pretende continuar (s/n) ");
      scanf("%c",&opcao);
      scanf("%c",&opcao);
      if ((opcao =='n') || (opcao == 'N'))
         printf("\n Resposta negativa!");
      else printf("\n Resposta positiva!");
         printf("\n\n Opção= %c \n\n",opcao);
   } while (opcao=='s');
   return 0;
}
```