Aula prática #5 – Estruturas de Controlo (Repetição)

Problema 1

Dois países A e B têm, respetivamente, P_a e P_b milhões de habitantes e taxas de crescimento anual (%) T_a e T_b. Assumindo que inicialmente P_a será sempre superior a P_b, e T_a será sempre inferior a T_b, desenvolva um programa que determine quantos anos serão necessários para que a população de B ultrapasse a de A.

Exemplo

```
Pais A (pop/taxa): 21 2
Pais B (pop/taxa): 15 3
Populacao de B ultrapassara a de A em 35 anos.
```

Problema 2

Escreva um programa que leia uma frase e indique quantas vezes foram escritas cada uma das vogais. Para implementar o programa utilize um ciclo de leitura de caracteres, terminando quando for encontrado o carácter ponto final '.'. Sugestão: utilize a instrução *switch*.

Exemplo

```
Qual e' a frase? A melhor forma de aprender a programar.
A - 6
E - 4
I - 0
O - 3
U - 0
```

Problema 3

Escreva um programa que simula n lançamentos de um dado (n definido pelo utilizador) e apresenta no final quantas vezes saiu a face seis. Sugestão: utilize a função *rand()*.

Exemplo

```
Quantos lancamentos? 10
A face seis saiu 2 vezes.
```

Problema 4

Escreva um programa que inverte a ordem dos dígitos de um número introduzido pelo utilizador.

Exemplo

```
Insira o numero: 123
O inverso do numero 123 e' 321
```

Problema 5

Implemente um programa que determine o valor de π utilizando o método de Monte Carlo http://en.wikipedia.org/wiki/Pi#Monte_Carlo_methods:

1. Considerar os contadores M e N para guardar o número de pontos dentro do círculo unitário e o número total de pontos, respetivamente.
2. Gerar um ponto aleatório, ou seja dois números reais x e y entre 0 e 1, usando por exemplo a instrução *rand() / (float)RAND_MAX*.
3. Se o ponto estiver dentro do círculo unitário $x^2 + y^2 < 1$, incrementar M.
4. Repetir passos 2 e 3 até ter sido gerado o número de pontos indicado pelo utilizador
5. Imprimir estimativa do π, dada por $\pi = 4M/N$.

Exemplo

```
100
pi: 4.000000
pi: 2.000000
pi: 2.666667
...
pi: 3.151515
pi: 3.120000
```

Problema 6

Com este exercício pretende-se descobrir e corrigir bugs num programa usando o **GDB**. Para tal iremos usar um programa que imprime os primeiros 10 números primos No entanto, devido à existência de bugs, o programa não desempenha corretamente essa função.

Nota: Escreva um programa que imprime os primeiros 10 números primos.

6.1 – Compile e corra o seguinte programa. Este programa imprime ":)" caso o programa se encontre correto e ":(" caso contrário. Conforme esperado, o programa deverá imprimir ":(".

```c
#include<stdio.h>

void testFunc(int n)
{
  if(n==29)
    printf(":)\n");
  else
    printf(":(\n");
}

int main()
{
  int n, i = 3, count, c;
  n = 10;

    printf("Os primeiros %d numeros primos sao:\n",n);
    printf("2\n");

   for ( count = 2 ; count <= n-1 ;  )
   {
      for ( c = 2 ; c <= i - 1 ; c++ )
      {
         if ( i/c == 0 )
            break;
      }
      if ( c == i )
      {
         printf("%d\n",i);
         count++;
      }
      i++;
   }
  testFunc(--i);
   return 0;
}
```

6.2 – Use o **GDB** para descobrir a localização dos bugs.

6.3 – Corrija os bugs. Depois de os bugs ficar devidamente corrigido, o programa deverá imprimir ":)".