Universidade Federal do ABC
Programação Estruturada
Fabrício Olivetti França
Lista de Exercícios 04

1 Objetivos da lista

Exercícios envolvendo aplicação de conceitos de structs, leitura e escrita de arquivos e bit tricks.

2 Exercícios

2.1 Structs

Para os seguintes exercícios utilize *scanf* para a leitura dos dados da seguinte forma:

```
scanf("%f, %f, %f", &(p1.x), &(p1.y), &(p1.z));
```

2.1.1 Geometria analítica

1. Crie uma struct para representar pontos em um espaço tridimencional (x, y e z, do tipo float). Com essa struct crie uma função que receba dois pontos com coordenadas em ponto flutuante (no formato (x, y, z) e calcule a distância entre dois pontos. A saída em ponto flutuante deve ser truncada em 2 casas decimais. Para calcular a distância entre dois pontos use a fórmula:

$$d_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - Y_A)^2 + (z_B - z_A)^2} \quad (1)$$

```
Entrada: 1, 2, 3
         3, 2, 1
Saida  : 2.82

Entrada: 4, 5, 6
         7, 8, 9
Saida  : 5.19
```

2. crie uma *struct* que represente um cronômetro definido por minutos, segundos e decimos, todos inteiros. Faça uma função que calcule e retorne a diferença entre duas medições de tempo (t2 - t1). Utilize *scanf("%dm %ds %d",...)* como formatação.

```
Entrada: 1m 10s 98
         2m 3s 32
Saida  : 0m 52s 34

Entrada: 2m 3s 22
         1m 12s 32
Saida  : -0m 50s 09
```

2.1.2 Lista de alunos

3. Crie uma struct para representar a ficha de um aluno (nome, número de matrícula, nota da P1, nota da P2, nota da P3), o nome terá no máximo 100 caracteres e a lista conterá 50 alunos, utilizando essa struct crie uma função que: leia do arquivo a lista de alunos (lista_alunos.txt) e calcule a média de cada aluno, imprimindo os resultados em outro arquivo com nome, número de matricula e média. Os pontos flutuantes devem ser limitados a 2 casas decimais. O conteúdo do arquivo de saída deve estar no formato do arquivo *lista_alunos.txt* fornecido. Para a leitura de cada linha utilize:

```
fscanf(fp, "\%s \%d \%f \%f %f",
           alunos[i].nome, &alunos[i].ra,
           &alunos[i].p1, &alunos[i].p2, &alunos[i].p3);
```

E para a escrita:

```
fprintf(fp, "\%s \%d \%f \%f %f %f\n",
           alunos[i].nome, alunos[i].ra,
           alunos[i].p1, alunos[i].p2, alunos[i].p3,
           alunos[i].media);
```

```
Entrada: arquivo lista_alunos.txt
Saida  : arquivo media_alunos.txt
```

2.2 Bit-Tricks

Utilize o seguinte código para os próximos exercícios, substituindo as funções *f1* e *f2* conforme requisitado:

```
#include <stdio.h>
#include <time.h>
#include <limits.h>

#define BIGNUM ULONG_MAX

typedef unsigned long int ulint;

ulint f1(ulint x)
{
  int div = 8;
  return x / div;
}

ulint f2(ulint x)
{
  return x >> 3;
}

int main(void) {

  clock_t tempo_init, tempo_fim;
  double tempo_gasto;
  ulint soma = 0;

  tempo_init = clock();
  for (int i=0; i<BIGNUM; i++) {
    soma += f1(i);
  }
  tempo_fim = clock();
  tempo_gasto = (double)(tempo_fim - tempo_init) / CLOCKS_PER_SEC;
  printf("Tempo gasto na versao normal: %lf\n", tempo_gasto);

  tempo_init = clock();
  for (int i=0; i<BIGNUM; i++) {
    soma += f2(i);
  }
  tempo_fim = clock();
  tempo_gasto = (double)(tempo_fim - tempo_init) / CLOCKS_PER_SEC;
  printf("Tempo gasto na versao bitwise: %lf\n", tempo_gasto);

  return 0;
}
```

Faça duas funções, uma com métodos convencionais e outra utilizando bit tricks para:

4. Checar se um número é ou não potência de 2.

5. É possível calcular a soma dos bits de um inteiro x somando o resto da divisão de x por 2 enquanto divide-se x por 2. Implemente a versão utilizando divisão e módulo ($f1$) e a versão utilizando operadores de bits ($f2$).

6. Receba dois números e diga qual o maior e qual o menor.