Aula Prática 7

Prazo de entrega: conferir no Moodle

Forma de Entrega: Enviar somente os arquivos `.c` e `.h` que você fez.

Exercício 1: Produto Escalar

Escreva um programa em C que recebe dois arranjos de números reais u e v e a dimensão n dos dois arranjos e que retorna o produto escalar de u e v. O produto escalar de dois arranjos é dado pela seguinte expressão:

$$u * v = u_0 * v_0 + v_1 * v_1 + ... + u_n * v_n.$$

Assuma que n é menor que o número máximo de elementos do arranjo (por exemplo, 100). Para testar, preencha cada vetor com um único valor.

Exercício 2: Fibonacci

A sequência de Fibonacci pode ser definida como:

fib(0) = 1
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2), para n>2

Implemente um programa que calcule a série de Fibonacci e armazene em um vetor cada elemento da série, de forma que a posição **0** armazene o termo **0**, a posição **1** o termo **1**, e assim por diante. Lembro que o termo **0** é o inteiro **1**. Seu programa deve receber do usuário um número entre **0** e **1000** e imprimir o termo correspondente ao número recebido. O usuário deve ser capaz de entrar com vários números interativamente em uma mesma execução. O programa termina quando o usuário entrar com um número **negativo** ou maior que **1000**. **Dica:** se o tipo ***int*** for insuficiente para armazenar todos os elementos da série, use outro tipo de dados.

Exercício 3: Média dos elementos de um vetor***

Escreva uma função em C que recebe um vetor de números reais v e número de elementos n armazenados em v e que retorna a média dos n elementos armazenados em v. O vetor deve ser preenchido com números aleatórios através de uma outra função. Para gerar números aleatórios, use a função `drand48()` no Linux, ou a função `rand()` no Windows, da biblioteca `stdlib.h`.

Exercício 4: Intercalação de vetores

Faça um programa que leia 2 vetores ***X*** e ***Y*** de 10 elementos, cada um. Intercale os elementos desses 2 vetores formando assim um novo vetor ***Z*** de 20 elementos, onde, nas posições ímpares de ***Z***, estejam os elementos de ***X*** e, nas posições pares, os elementos de ***Y***. Exemplo: Se ***X = 3,5,2,8,4*** e ***Y = 1,7,6,5,2*** então ***Z = 3,1,5,7,2,6,8,5,4,2***. Imprimir o vetor ***Z***.

Exercício 5: Inverso de um vetor

Faça um programa para ler um vetor X de n elementos e gerar um outro vetor com esses n elementos em ordem inversa. Exemplo: Se $X = 3, 5, 2, 8, 4$, deverá ser gerado um vetor $Y = 4, 8, 2, 5, 3$. O valor de n é lido pelo teclado.

Exercício 6: União de vetores

Faça um programa para preencher dois vetores X e Y de 60 posições com valores aleatórios entre 0 e 365. Imprima o conjunto união desses dois vetores. Curiosidade: leia sobre o paradoxo do aniversário.

Exercício 7: Teste da Função `rand()`

Uma boa função para geração de números aleatórios deve gerar valores com igual probabilidade, i.e., se eu quero gerar 100 valores entre 1 e 10, o número de vezes que cada número é gerado deve ser próximo de 10. Assim, nesta prática você deve criar um programa para testar a qualidade da função `rand()` da linguagem C. Para isso, gere $5.000.000$ números aleatórios entre 0 (inclusive) e 999 (inclusive) e conte quantas vezes cada número foi gerado. Imprima a diferença entre a maior e a menor contagem. Exemplo: se o número 83 foi aquele que mais vezes foi gerado, com 5315 gerações, e o número 762 foi aquele que menos vezes foi gerado, com 4802 gerações, então seu programa deve imprimir $5315 - 4802 = 513$.

Dica: crie um vetor de inteiros de 1000 posições para armazenar quantas vezes cada número (entre 0 e 999) é gerado. Assim, cada vez que você gerar aleatoriamente o número k, faça `vetor_contagem[k]++`. Exemplo: se você gerar o número 888, faça `vetor_contagem[888]++`. No final, `vetor_contagem[k]` conterá o número de vezes que o número k foi gerado.

DESAFIO PARA OS FORTES: Além da diferença, imprimir também o desvio padrão das contagens. Para entender o que é o desvio padrão e como ele é calculado, consulte a Wikipedia ou qualquer site disponível na Internet.

Teoria

Números aleatórios

A única função que temos na linguagem C para gerar um número aleatório é a função \texttt{rand}. Essa função não requer parâmetros e retorna um número inteiro aleatório entre 0 e $2^{31} - 1$ (maior inteiro possível, lembram?). Exemplo de código que gera três números aleatórios:

```
#include <stdio.h>

void main() {

      int r1, r2, r3;
      r1 = rand();
      r2 = rand();
      r3 = rand();

      printf("numeros gerados: %d, %d, %d", r1, r2, r3);
}
```

Vetores

Para calcular a média dos 100 números inteiros gerados aleatoriamente do Problema 1, você precisa de 100 variáveis inteiras. Com o conhecimento que temos hoje, teríamos que criar 100 variáveis, por exemplo, $r1, r2, ..., r100$, o que é impraticável. Assim, para resolver esse problema, podemos criar uma única variável do tipo *vetor* para armazenar todos esses valores. Essa variável do tipo *vetor* pode ter, por exemplo, o nome r e ter associada a ela 100 endereços de memória para guardar inteiros. Como eu faço isso?

```
int r[100]; //cria um vetor de inteiros com 100 posicoes
```

Assim, eu posso considerar que cada uma das 100 posições é uma variável única para eu armazenar um valor inteiro. E como eu acesso essas variáveis? Basta eu utilizar o índice da posição que eu quero acessar entre colchetes, depois do nome da variável. Por exemplo, para eu armazenar o número -15 na primeira posição do vetor `r` e o número 948 na quadragésima posição de `r`, devo executar as seguintes operações:

```
r[0] = -15;
r[39] = 948;
```

Note que a primeira posição de um vetor de 100 posições é 0 e a última posição é 99. Então, como faço para calcular a média desses dois números?

```
float media = (r[0] + r[39])/2.0;
```

O código completo é:

```
#include <stdio.h>

void main() {

      int r[100]; //cria um vetor de inteiros com 100 posicoes
      r[0] = -15;
      r[39] = 948;
      float media = (r[0] + r[39])/2.0;
}
```