Universidade Federal de Viçosa
Campus Rio Paranaíba
SIN 110 - Programação

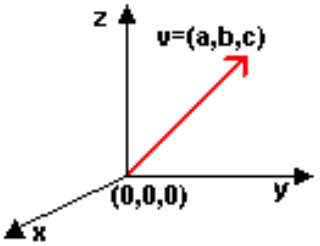
Instruções

- Entregar, impreterivelmente, na aula de 20 de maio.
- **NÃO SERÃO ACEITAS ENTREGAS FORA DO PRAZO!!!**
- A lista deve ser realizada individualmente e manuscrita.
- Boas práticas de programação e **organização** na entrega compõem a nota.

1. Considere a estrutura:

```
struct Vetor{
   float x;
   float y;
   float z;
};
```

De posse da estrutura, implemente uma função que represente dois vetores no espaço R^3. Um exemplo de vetor v, no espaço R^3, é ilustrado pela figura abaixo.



Em seguida, a função criada deve calcular a soma desses dois vetores. Tal função deve possuir o seguinte protótipo:

```
void soma(struct Vetor *v1, struct Vetor *v2,
          struct Vetor *res);
```

onde os parâmetros *v1* e *v2* referem-se aos vetores a serem somados, e o parâmetro *res* refere-se a estrutura onde o resultado da operação deve ser armazenado. Ao final, imprima o os elementos do vetor res na função main, onde foi feita a chamada da função soma.

2. Escreva um programa que simule o lançamento de uma moeda. Para cada lançamento, o programa deve imprimir **Cara** ou **Coroa**. Deixe o programa lançar a moeda 100 vezes e conte quantas vezes cada lado da moeda foi sorteado. Por fim, imprima os resultados na tela.
   Seu programa deve:
   - Chamar a função *lancaMoeda()*, a qual não utiliza argumentos e retorna **0** para **Cara** e **1** para **Coroa**.
   - Utilizar a função *rand()* para simular o lançamento da moeda.

3. Sejam a e b os catetos de um triângulo, onde a hipotenusa é obtida pela equação: $hipotenusa = \sqrt{a^2 + b^2}$. Faça um programa que contenha uma função que receba os valores de a e b para calcular o valor da hipotenusa através da equação.

4. Codifique, compile e execute um programa que leia os elementos de uma matriz inteira de tamanho 5x5 e imprima: (i) os elementos da diagonal secundária e (ii) a soma dos elementos da diagonal secundária.

Exemplos de entrada e saída

Entrada	Saída
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 5
1 1 1 1 1 5 4 3 2 1 6 7 8 9 0 0 9 8 7 6 1 2 3 5 7	1 2 8 9 1 21

5. Codifique, compile e execute um programa que:
   - Contenha uma struct para representar um novo tipo Pessoa, o qual contém: Nome, Altura e Peso.
   - Possua uma função sem retorno que calcula e imprime o IMC de uma pessoa, sendo que essa função recebe como parâmetro apenas uma variável do tipo pessoa.

   Lembre-se: IMC = Peso/altura²
   - Crie uma variável do tipo Pessoa capaz de armazenar 30 indivíduos e entre com os dados de cada uma deles.
   - Por fim, chame a função de cálculo do IMC criada para cada uma das pessoas.

6. Utilizando a passagem de parâmetro por referência, crie as seguinte funções:
   - `void preencheVetor(int *vet, int n);`

- `void imprimeVetor(int *vet, int n);`
- `int retornaMaiorElemento(int *vet, int n);`

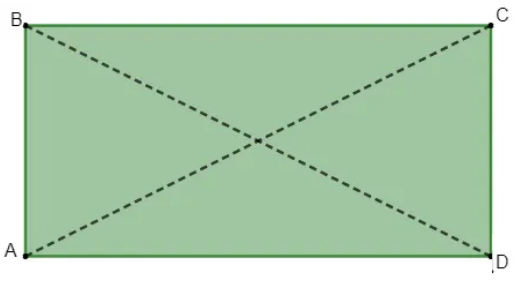
7. Faça um programa para simular uma agenda de telefones. Para cada pessoa devem-se ter os seguintes dados:
    - Nome
    - E-mail
    - Endereço (contendo campos para rua, número, complemento, bairro, cep, cidade, estado e país)
    - Telefone (contendo campo para DDD e número)
    - Data de aniversário (contendo campo para dia, mês, ano).
    - Observações: Uma linha (string) para alguma observação especial.

a) Definir as estruturas acima.
b) Declarar a variavel agenda (vetor) com capacidade de agendar até 100 contatos.
c) Definir um bloco de instruções busca por primeiro nome: Imprime os dados da pessoa com esse nome (se tiver mais de uma pessoa, imprime para todas).
d) Definir um bloco de instruções busca por mês de aniversário: Imprime os dados de todas as pessoas que fazem aniversario nesse mês.
e) Definir um bloco de instruções busca por dia e mês de aniversário: Imprime os dados de todas as pessoas que fazem aniversario nesse dia e mês.
f) Definir um bloco de instruções insere pessoa: Insere por ordem alfabética de nome.

8. Considere a seguinte estrutura:

```
struct Ponto{
   int x;
   int y;
};
```

A fim de representar um ponto em uma grade 2D, implemente uma função que indique se um ponto p está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo (v1) e superior direito (v2). O vértice de um retângulo consiste no ponto de encontro de dois lados.
Por exemplo, na figura abaixo, v1 está sendo representado pelo ponto A e v2 está sendo representado pelo ponto C.



A função deve retornar 1 caso o ponto esteja localizado dentro do retângulo e -1 se o ponto estiver fora do retângulo. A sua função deverá obedecer o seguinte protótipo:

```
int dentroRet(struct Ponto *v1, struct Ponto *v2, struct Ponto *p);
```