CIV 2802 – Sistemas Gráficos para Engenharia – 2011.1

2º Trabalho: Programação básica em C++: Funções

Implementação de programas elementares

Entrega: 22/março/2011

Pede-se as implementações das funções descritas a seguir. Para cada um delas, deve-se construir um programa (função `main`) para testar a implementação, lendo e imprimindo parâmetros, se necessário.

Item 1

Implemente uma função que calcule o número de combinações de *n* elementos tomados *k* a *k*. Esse número é dado pela fórmula do arranjo:

$$a = \frac{n!}{(n-k)!}$$

A função deve ter como valor de retorno o arranjo calculado, obedecendo ao protótipo:

```
int arranjo (int n, int k);
```

O cálculo do fatorial deve ser implementado através de uma função.

Item 2

Implemente uma função que indique se um ponto (*x*,*y*) está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo (*x0*,*y0*) e superior direito (*x1*,*y1*). A função deve ter como valor de retorno `1`, se o ponto estiver dentro do retângulo, e `0` caso contrário, obedecendo ao protótipo:

```
int dentro_ret (int x0, int y0, int x1, int y2, int x, int y);
```

Item 3

Implemente uma função que calcule as raízes de uma equação do segundo grau do tipo $ax^2+bx+c=0$. Essa função deve retornar os valores das raízes através dos parâmetros *x1* e *x2*, cujos endereços em memória são passados para a função. O protótipo da função é:

```
int raizes (float a, float b, float c, float *x1, float *y2);
```

Essa função deve ter como valor de retorno o número de raízes reais e distintas da equação. Se existirem raízes reais, seus valores devem ser armazenados nos parâmetros *x1* e *x2*.

Item 4
Implemente uma função que receba como parâmetro um vetor de números reais (`vet`) de tamanho `n` e retorne quantos números negativos estão armazenados nesse vetor. Essa função deve obedecer ao protótipo:

```
int negativos (int n, float *vet);
```

O programa principal (`main`) deve pedir para o usuário entrar com o tamanho do vetor, alocar memória dinamicamente para o vetor de acordo com o tamanho fornecido, e pedir para o usuário entrar com os valores do vetor.

Item 5
Implemente uma função que receba como parâmetro um vetor de números inteiros (`vet`) de tamanho `n` e inverta a ordem dos elementos armazenados nesse vetor. Essa função deve obedecer ao protótipo:

```
int inverte (int n, int *vet);
```

O programa principal (`main`) deve pedir para o usuário entrar com o tamanho do vetor, alocar memória dinamicamente para o vetor de acordo com o tamanho fornecido, e pedir para o usuário entrar com os valores do vetor.

Item 6
Implemente uma função que permita a avaliação de polinômios. Cada polinômio é definido por um vetor que contém seus coeficientes. Por exemplo, o polinômio de grau 2, $3x^2+2x+12$, terá um vetor de coeficientes igual a . A função deve obedecer ao protótipo:

```
double avalia (double *poli, int grau, double x);
```

Onde o parâmetro `poli` é o vetor com os coeficientes do polinômio, `grau` é o grau do polinômio, e é o valor para o qual o polinômio deve ser avaliado.

O programa principal (`main`) deve pedir para o usuário entrar com o grau do polinômio, alocar memória dinamicamente para o vetor dos coeficientes de acordo com o grau fornecido, e pedir para o usuário entrar com os valores dos coeficientes do polinômio.