BCC202 – Estrutura de Dados I

2s2015

Prática 03: Recursão

Exercícios

1. Seja um vetor v de inteiros de tamanho n. Queremos saber a soma de todos os seus elementos. Como podemos descrever este problema de forma recursiva? Implemente a função correspondente.

2. Implemente de forma recursiva uma função que permita somar dois números inteiros não negativos, x e y, usando apenas incrementos (++) e decrementos unitários (- -).

3. O algoritmo de Euclides para o cálculo do Máximo Divisor Comum entre dois números inteiros são negativos x e y pode ser resumido na seguinte fórmula:

$$mdc(x,y) = \begin{cases} x, & \text{se } y = 0 \\ mdc(y, x \bmod y), & \text{se } y > 0 \end{cases}$$

Implemente o MDC de forma **iterativa** e **recursiva**.

4. Implemente uma função que permita encontrar o maior elemento de um vetor de inteiros positivos de forma recursiva.

5. Implemente de forma recursiva uma função que permita calcular o valor de K^n, onde n é um número inteiro não negativo e k é um número real. Escreva e resolva a equação de recorrência dessa função. Qual é a ordem de complexidade da sua função? Implemente de forma iterativa a função para calcular K^n. Qual a versão é mais eficiente, iterativa ou recursiva? Justifique sua resposta.

6. A pesquisa ou busca binária (em inglês, "*binary search algorithm*") é um algoritmo de busca em vetores que segue o paradigma de "divisão e conquista". Ela assume que o vetor está ordenado e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso retornando o índice da posição da chave. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua recursivamente na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua recursivamente na metade inferior do vetor.

a- Implemente a busca binária usando uma função recursiva, cuja assinatura é ilustrada a seguir.

```
// Esta função recebe um inteiro “x”, um vetor ordenado v[0..n-1], um índice para
// a posição inicial do vetor “e”, e posição final “d”. A função devolve um índice
// “m” em 0..n-1 tal que v[m] == x. Se tal m não existe, devolve -1.

int buscaBinaria( int x, int v[], int e, int d);
```

b- Formule e resolva a equação de recorrênncia do seu algoritmo e defina a ordem de complexidade da sua função.

IMPORTANTE: No método main(), implemente chamadas para testar todas as funções implementadas para as questões descritas acima.

Procedimento de Entrega

Você deve entregar os exercícios no Moodle até o dia **29/11/15** até às **23:55**. Não serão aceitas soluções entregues após o horário estipulado.

Instruções gerais:

1- Crie uma pasta para a Prática 03: PrimeiroNome-UltimoNome-pratica03 (exemplo: amanda-nascimento-1).
2- Crie um arquivo .h contendo a protótipo de todas as funções a serem implementadas.
3- Crie um arquivo .cpp contendo a implementação dos protótipos previamente definidos.
5- Crie um arquivo .cpp que irá contem o método main(), que deve conter os testes para as funções implementadas.
4- Compile na linha de comando usando g++ *.cpp -o prog.exe.
5- Apague os arquivos gerados na compilação (mantenha apenas os arquivos .cpp e .h).
6- Quando necessário, crie um arquivo “.doc” com respostas adicionais (e.g., equação de recorrência e ordem de complexidade de uma função).

Compacte a pasta criada anteriormente num único arquivo (“*.zip”*), que deve ser entregue via Moodle. ***O arquivo compactado deve também conter um arquivo “.txt” com seu nome e número de matrícula.***