UM EXPERIMENTO DE COMPARAÇÃO DE TEMPOS DE EXECUÇÃO DE ALGORITMO DE CLASSIFICAÇÃO BASEADO NO MÉTODO DA INSERÇÃO DIRETA IMPLEMENTADOS NAS LINGUAGENS JAVA E C

[1] BIM, Pedro Henrique Marana; [1] RABALDELLI,Diego José Caíres; [1] SOARES, Luis Antonio; [2]TAMAE, Rodrigo Yoshio; [2]MUZZI, Fernando Augusto Garcia; [2]ROSA, Adriano Justino

1 - Discentes do Curso Sistemas de Informação - FAEG/Garça
2 - Docente do Curso Sistemas de Informação - FAEG/Garça
pedrohmbim@yahoo.com.br; rytamae@yahoo.com.br; fagmuzzi@yahoo.com.br

RESUMO

Caracterizado como duas linguagens extremamente relevantes no ambiente de desenvolvimento de softwares, além de seu intrínseco grau de relação, as linguagens C e Java, apresentam-se neste trabalho como objetos comparativos, levando-se em conta seus respectivos tempos de execução, gerados sob as mesmas circunstâncias, com fins estatísticos de comparação de tempos de execução, nos permitindo afirmar comprovadamente a excelência da linguagem C, através de dados irrefutáveis.

Palavras-chave: Java, C, comparação de tempo de execução.

ABSTRACT

Characterized as two extremely excellent languages in the environment of development of softwares, beyond its intrinsic degree of relation, the languages C and Java, are presented in this work as comparative objects, taking in account its respective times of execution, generated under the same circumstances, with statistical ends of comparison of execution times, in allowing provly to affirm the excellency of language C, through irrefutable data.

Keywords: Java, C, time of execution comparison.

1 - INTRODUÇÃO

O uso das mais diversas linguagens de programação tem por objetivo resolver problemas específicos ou prover aos profissionais de tecnologia, entre outras coisas, desempenho suficiente para desenvolver soluções ágeis e confiáveis. A escolha por utilizar uma linguagem, hoje, requer a análise de parâmetros baseados em apontamentos relacionados aos fins de cada projeto.

Atualmente, considera-se de suma importância uma fundamentação e o entendimento prático de Algoritmos e Estruturas de Dados, pois possibilita agregar funções que permitam a comparação do tempo de execução entre o método da inserção direta, executado em linguagens C e Java.

Ainda no âmbito de estrutura de dados, os métodos de ordenação são meios extremamente importantes e necessários para classificação de dados, em vetores, uma vez que possibilitam menos esforços, otimizando o processamento do software e máquina em questão, para a localização de dados, ou seja, o objetivo da ordenação é facilitar a localização dos membros de um conjunto de dados, e é uma

atividade fundamental e universalmente utilizada para a elaboração de algoritmos mais complexos. Paralelamente, a medição do tempo de execução permite comparação das linguagens, permitindo traçar parâmetros de uso e melhorias, além de comprovar praticamente fundamentação teórica, através das possibilidades e limitações de cada linguagem.

Paralelamente, o método da inserção direta, é caracterizado segundo PASSARO, 2006 como o mais eficaz dos considerados básicos (bubblesort, inserção direta e seleção direta), o método da inserção direta baseia-se, conceitualmente segundo (PASSARO, 2006), em uma classificação onde inicialmente, considera-se o primeiro elemento ordenado, o segundo elemento é, então, inserido na sua posição correta em relação ao primeiro, resultando as duas primeiras posições ordenadas. A seguir, o terceiro elemento é inserido na sua posição correta em relação aos dois primeiros, resultando nas três primeiras posições ordenadas e assim sucessivamente.

Dentre os métodos de inserção (inserção direta e shell), o método da Inserção Direta é o mais simples, porém, é o mais rápido**,** entre os outros métodos considerados básicos – Bubblesort e Seleção Direta (PASSARO, 2006). Neste algoritmo, o próprio vetor é utilizado no processo de ordenação, não consumindo, portanto, memória para a separação dos segmentos do vetor, consome-se um pouco de memória somente para o armazenamento de variáveis auxiliares (PASSARO, 2006). Todavia, a eficácia do método de inserção está intrinsecamente ligada a uma adequação aos seguintes fatores: o número de registros a serem classificados, se todos os registros caberão ou não na memória interna disponível, o grau de classificação já existente;

No entanto, a ordenação por inserção tem duas vantagens. Primeiro, ela se comporta naturalmente, ou seja, trabalha menos quando a matriz já está ordenada e o máximo quando a matriz está ordenada no sentido inverso. Isso torna a ordenação por inserção excelente para listas que estão quase em ordem. A segunda vantagem é que ela não rearranja elementos de mesma chave. Isso significa que um vetor que é ordenado por duas chaves permanece ordenado por ambas as chaves após uma ordenação por inserção (CCET-VIRTUAL, 2006).

2 - APLICAÇÕES : IMPLEMENTAÇÃO E DESENVOLVIMENTO

Baseando-se nos conceitos dos algoritmos de classificação, especificamente no método da inserção direta, a comparação de tempo entre as aplicações desenvolvidas em linguagens C e Java, foi executada utilizando IDE em C, Dev-C++, versão 4.9.8.0, de junho de 1991, e em Java , JCreator, versão 3.50.009 de 2005. A máquina utilizada para testes possui a seguinte configuração: AMD Athlon (TM) XP 1700+ 1,47 GHz, 128 MB de RAM, onde foram realizadas 30 execuções de cada programa, sendo gerado 5000 mil números aleatoriamente e ordenados através do método da inserção direta. O desenvolvimento da aplicação em C, estruturou-se na função que gera 5000 números aleatórios através da função *rand()* presente na biblioteca *stdlib.h*, e evidenciados na figura 1, pela cor azul, que posteriormente serão ordenados através do método da inserção direta, evidenciado no código pelas cores verde (a função de inserção direta) e rosa (quando a função principal o chama para a ordenação), e os comando de medição de tempo que fazem a marcação inicial antes da geração dois números e após a ordenação, sendo apresentado à diferença entre a primeira e a última tomada de tempo realizada pela biblioteca *time.h*, através da função *difftime* e que estão e sendo referenciados pela cor vermelha.

```
# include <stdio.h>
# include <time.h>
# include <stdlib.h>
int insercao (int n, int* v);
   int main (void)   {
        int i, m;
        int v [5000];
        time_t inicio, fim;
        printf("Gerando numeros aleatorios com rand\n");
        time(&inicio);
        for(i=0;i<= 5000; i++) {
           v[i]=rand();
           printf("o valor %d\n", rand());}
        insercao (5001,v);
        for (m=0; m<=5000; m++) {
         printf ("eis o vetor  %d\n ", v [m]);
              } time(&fim);
              printf ("\n\n demorou: %10.10f segundos\n",difftime(fim,inicio));
              getchar (   );
          }
         int insercao(int n, int * v)
         {
            int aux, i, j;
          for (j=1 ; j<n; j++)
           {
            aux = v[j];
            i= j -1;
           while ((i>=0 )&& (v[i]> aux))
             {
              v[i + 1]= v [i];
              i--;
             }
          v[i + 1]= aux;

            }
         }
```

Figura 1- Código da aplicação em linguagem C

A seguir a tabelas 2, está apresentando de forma separadas para fins didáticos, entretanto ambas estão dentro da classe *insercaodireta* (linhas 5 a 55). A segunda aplicação implementada em linguagem Java, apresenta a mesma estrutura acima descrita, onde a geração dos 5000 números aleatórios é realizada, como pode ser visto abaixo na Tabela 2 (linhas 19 a 23), onde se refere aos métodos da classe *Random* (linha 2), e posteriormente o vetor é ordenado, após chamar-se o método da inserção (linha 25), que refere-se a Tabela 3, ao método inserido na classe (linhas 37 a 54), e a medição é realizada, mediante métodos da classe *Date* (linha 3), declarada na classe *insercadireta*, onde na primeira tomada (linha 13) e a segunda (linha 32), para que seja realizada a diferença (linha 33) que será apresentada (linha 34).

```
1  import java.io.*;
2    import java.util.Random;
3    import java.util.Date;
4
5    class insercaodireta
6    {
7       public static void main (String args[])
8       {
9            long inicio,end, diff;
10           Random rand = new Random();
11           System.out.print("Implementacao do algoritmo de insercao em Java\n\n");
12
13           inicio = System.currentTimeMillis();
14
15           System.out.print("\nEis os numeros aleatorios\n\n");
16
17           int vet[]= new int [5000];
18
19           for (int i=0;i<vet.length; i++)
20           {
21           vet[i]= rand.nextInt(50000);
22           System.out.println ("o valor e " +vet[i]);
23           }
24
25           insert(vet);
26           System.out.println ("\n\nVetor ordenado por insercao \n\n");
27
28           for (int i = 0; i < vet.length; i++)
29            {
30             System.out.println ("eis o vetor "+vet[i] );
31            }
32            end = System.currentTimeMillis();
33            diff = end - inicio;
34            System.out.println("Demorou " + (diff / 1000.0) + " segundos");
35      }
```

Figura 2- Método principal contendo os comandos para geração de números, ordenação e medição de tempo, dentro da classe *insercaodireta.*

Concomitantemente, apresentados os códigos e suas respectivas explicações, serão apresentados a seguir as comparações de tempo executadas nas duas linguagens, demonstrando os números que nos permitem afirmar, a superioridade da linguagem C.

Mediante a comparação das tomadas de tempo das implantações em Java e C, podemos demonstrar que apesar de ser mais eficiente, a linguagem C, mostrou um intervalo mais amplo (variando entre 4 e 10 segundos), apesar de ser mais centralizado; Java apesar de ser mais variado, possui um intervalo menor (variando

entre 4 e 8 segundos). Paralelamente, tendo como base gráfico apresentado na Tabela 7, podemos comprovar a relativa estabilidade de C, que mesmo com freqüência visivelmente superior em 4 segundos, apresenta-se estável nos demais; todavia Java apresenta dois pontos de pico, sendo o principal em 8 segundos, demonstrando que apesar de possuir um tempo maior de execução teve uma distribuição homogênea de variação de tempo.

Assim realizado, as tomadas de tempo das implementações em C e Java comprovam praticamente a excelência de C, algo que pode ser explicado por uma desvantagem de Java que pode ser parcialmente entendida, se levarmos em consideração que tratamos de uma linguagem de programação realmente multi-plataforma, graças ao JVM (*Java Virtual Machine*), que é a responsável pela execução de um programa escrito em Java. Um código fonte escrito em Java é traduzido para bytecode, que é lido e executado pela JVM. Esse processo de compilação e execução possibilita a vantagem da linguagem C, mas isso não impede que o Java seja implantado com qualquer tipo de software.

5 - CONCLUSÃO

Apresentadas como duas linguagens extremamente importantes na área da programação e com estruturas relativamente similares, devido à suas histórias de co-irmãs, este experimento demonstrou fundamentos técnicos das linguagens, somadas a fundamentação prática e teórica dos algoritmos de classificação, alem de apresentar dados comprovados da eficácia das duas implementações.

A implementação fundamentou-se em na tríade lógica-conceito-prática, das linguagens, dos algoritmos de classificação , particularmente do método da inserção direta, onde o experimento de comparação estabeleceu-se em 30 tomadas de tempo, previamente anotadas e em ambientes iguais, de duas implementações em C e Java, com estruturas idênticas, onde foram gerados 5000 mil números aleatórios, que foram ordenados através do método da inserção direta, com tomadas de tempo antes da geração dos números e após a ordenação, sendo usado a diferença entre as tomadas. Desta forma o experimento apresentou a superioridade, na prática, da linguagem C, que possuiu tempos de execução menores, em média 12%. Este excesso temporal de Java, analogicamente transposto para elaboração

de sistemas complexos concorrentemente a C, pode inviabilizar o seu uso apesar de suas demais vantagens e qualidades tão difundidas e expostas. A desvantagem de Java, pode ser entendida graças ao JVM (Java Virtual Machine) , que é a responsável pela execução de um programa escrito em Java. Um código fonte escrito em Java é traduzido para bytecode, que é lido e executado pela Java Virtual Machine, possibilitando assim um tempo maior de execução e vantagem para C. Entretanto esta desvantagem não desmerece a linguagem Java que possui fundamentação de mercado, é de propósito geral, além de ser um dos símbolos da orientação à objetos, mas somente apresenta falhas que podem proporcionar uma evolução, possibilitando avanços maiores a esta linguagem tão difusa aceita pelo campo da programação.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

CARDOSO, Aníbal P. **Métodos de Ordenação**. Disponível em: http://www.inf.unisinos.br/~anibal/prog2ordena.pdf. Acessado em:15/08/2006.

CCET-Virtual (Centro de Ciências Exatas e Tecnológicas – UCS). **Vetor** .Disponível em: http://ccet.ucs.br/dein/napro/java/materialapoio/vetor/vetor.html. Acessado em:18/08/2006.

CEFETPB,Classificação de **Dados-Métodos de Classificação** Interna.Disponível em: http://arquivos.coinfo.cefetpb.edu.br/~fred/ped/material/apostilas/PED-Classif.pdf. Acessado em: 18/08/2006.

CORMEN, H.Thomas; Leiserson, E.Charles; Rivest, L.Ronald; Stein,Clifford. **Algoritmos Teoria e Prática** .Tradução:Vandenberg D. de Souza.Tradução da 2ª edição americana, páginas 11 a 21.Editora Campus 916 p.Rio de Janeiro, 2002.

GONÇALVES, Ronaldo A. L. **Algoritmos e Estruturas de Dados - Técnicas de Programação Usando a Linguagem Pascal**.Disponível em: http://www.din.uem.br/~ronaldo/LivroAED-Capitulos-1-2-3- IntrodPascal.pdf. Acessado em 14/09/2006.

PÁSSARO, Ângelo**.Projeto e Análise de Algoritmos: Algoritmos de Ordenação** Disponível em: http://www2.brazcubas.br/Algoritmos_de_Ordenacao.pdf. Acessado em :18/08/2006