Aula prática 11

Esta aula tem como objetivo fazer uma comparação entre as várias estruturas estudadas, para verificar qual a estrutura que se porta melhor a inserir elementos e a pesquisar elementos (vector, listas, arvores e tabelas de dispersão).

1 Utilizar o código dos trabalhos práticos com as classes User, TVSeries, TVSeriesManagement, UserManagement, TVSeriesManagementList, UserManagementList, NodeUser, UserManagementTree, HashTable para verificar o tempo que demora a inserir elementos.

1.1 Utilize o seguinte código para medir os tempos:

```
clock_t beg, end;
double time;
beg = clock();

 Código da tarefa que pretendemos medir


end = clock();
time = (double)(end - beg) / CLOCKS_PER_SEC;
cout << "Time is "<< time << endl;
```

Adapte o seguinte código para criar novos User(com valores aleatórios) para cada estrutura(elementos a serem inseridos):

```
int total=1000;
User* users;
.
vector<int> num;
int random;
while (num.size()!=total)
{
    random = rand() % total;
    if (find(num.begin(),num.end(),random)==num.end())
    {
        num.push_back(random);
    }
}

Código inicial para medir tempo


for (int i=0;i<total;i++)
{
    us="user"+to_string(num[i]);
```

```
    users = new User(us, us, "P", {});
   código de inserir elemento numa estrutura

}
Código final para medir tempo
```

Exemplo para inserir no fim (primeiro espaço livre do vetor) da estrutura vector do STL, usando a classe UserManagement.

```
int total=1000;
User* users;
vector<int> num;
int random;
while (num.size()==total)
{
    random = rand() % total;
    if (find(num.begin(),num.end(),random)==num.end())
    {
        num.push_back(random);
    }
}

clock_t beg, end;
double time;
beg = clock();
for (int i=0;i<total;i++)
{
    us="user"+to_string(num[i]);
    users = new User(us, us, "P", {});
    userMVector.addUser(users);
}
end = clock();
time = (double)(end - beg) / CLOCKS_PER_SEC;
cout << "Time is "<< time << endl;
```

Pretende-se que preencha a seguinte tabela com os tempos para a inserção de (1000 , 5000, 10000, 50000, 100000) elementos em cada uma das estruturas:

N° elementos	Vector no fim	Vetor no início	Lista no fim	Lista no início	arvores	Tabela de dispersão
1000						
5000						
10000						
50000						
100000						

Notas:
Para cada valor que preencher na tabela faça a média de 3 tentativas.

Para alterar o número de elementos, modifica-se a variável total.

Para adicionar no início é preciso ir modificar a função addUser de cada class para inserir no início.

Para a tabela de dispersão deve-se usar logicamente o countryStats em vez do user.

1.2 Utilize o seguinte código para utilizar vetores sem usar a biblioteca STL, para verificar a inserção de elementos no inicio

```
int total=1000;
User* users;
User* userVet[total];
clock_t beg, end;
double time;
beg = clock();
for (int i=0;i<total;i++)
{
    us="user"+i;
    users = new User(us, us, "P", {});
    for(int j=i-1;j>=0;j--)
    {
        userVet[j+1]=userVet[j];
    }
    userVet[i]=users;
}
end = clock();
time = (double)(end - beg) / CLOCKS_PER_SEC;
cout << "Time is "<< time << endl;
```

N° elementos	Vector no inico sem STL
1000	
5000	
10000	
50000	
100000	

2. Preenche a tabela com os tempos de pesquisar nas diversas estruturas. Insera os elementos em cada estrutura e depois contabilize o tempo de fazer a pesquisa de todos os elementos que inseriu.

N° elementos	Vector	Lista	arvores	Tabela de dispersão
1000				
5000				
10000				
50000				
100000				

3. Faça um gráfico para cada uma das tabelas.