# 2022_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA_TN - METATURMA



## L01E06 - Folha Salarial (4,0 pts)

**Data de entrega**: terça, 17 Mai 2022, 23:59
**Arquivos requeridos**: main.cpp, Jogador.hpp, Jogador.cpp, Time.hpp, Time.cpp, RegistroNBA.hpp, RegistroNBA.cpp (Baixar)
**Tipo de trabalho**: Trabalho individual

O objetivo deste exercício é praticar o uso de diferentes estruturas da **STL**.

Você deve fazer um programa para ajudar na gestão da folha salarial de times da NBA. Para isso, deve-se implementar três TADs: **Jogador**, **Time** e **RegistroNBA**. Cada TAD deve seguir as especificações abaixo:

- **Jogador:**

  1. Atributos: string _nome, _posicao; int _salario;
  2. `Jogador(string nome, string posicao, int salario)`: Método construtor para a inicialização dos atributos.
  3. `void imprimir_dados()`: faz a impressão dos atributos no seguinte formato: "**nome posicao salario**", com uma quebra de linha ao final. Atenção, nesse caso utilize **tab** (\t) para separar os elementos.

- **Time:**

  1. Atributos: string _nome; e _jogadores (que guarda uma coleção de elementos do tipo Jogador, você deve escolher a estrutura que julgar mais adequada para usar).
  2. `Time(string nome)`: método construtor para a inicialização dos atributos.
  3. `void adicionar_jogador(string nome, string posicao, int salario)`: adiciona um novo jogador na coleção.
  4. `void imprimir_lista_jogadores()`: imprime o nome do time, seguido das informações dos jogadores do time, ou seja, percorre toda a lista e chama o método 'imprimir_dados()'. A lista deve ser impressa **ordenada de maneira descendente** (maior para o menor) de acordo com os **salários dos jogadores**.
  5. `void imprimir_folha_salarial_consolidada()`: imprime o valor total da folha salarial do time, ou seja, o somatórios dos salários individuais. A impressão é no formato: "**NomeTime ValorFolha**". Utilize **tab** (\t) para separar os elementos.

- **RegistroNBA:**

  1. Atributos: crie uma variável para guardar as informações dos times. Você deve escolher a estrutura que julgar mais adequada para usar, lembrando que deve ser fácil acessar cada time. Dica: um map é bem útil!
  2. `void adicionar_time(string nome)`: adiciona um novo time na coleção. Você pode assumir que os nomes dos times serão únicos e formados por apenas uma palavra.
  3. `void adicionar_jogador(string nome_time, string nome_jogador, string posicao, int salario)`: adiciona um novo jogador a um determinado time de acordo com os parâmetros informados. Atenção: se o time informado não existir na coleção, ele deve ser inserido nesse momento.
  4. `void imprimir_lista_jogadores_time(string nome_time)`: imprime as informações dos jogadores do time informado.
  5. `void imprimir_folha_consolidada_time(string nome_time)`: imprime o valor total da folha salarial do time informado.
  6. `void imprimir_folha_salarial_geral()`: imprime a folha salarial consolidada de todos os times registrados. A impressão deve ser feita em **ordem alfabética** pelos nomes dos times.

Você é livre para adicionar nos TADs quaisquer outros atributos ou métodos auxiliares que julgar necessário.

Por fim, você deve implementar o arquivo **main.cpp** e adicionar toda a parte de entrada/saída que será responsável por manipular os seguintes

comandos:

- '**`t nome_time`**': comando para adicionar um time.
- '**`j nome_time nome_jogador posicao salario`**': comando para adicionar um jogador. Você pode assumir que todas as strings *não* possuem espaço, ou seja, são palavras únicas.
- '**`l nome_time`**': imprime a lista de jogadores de um determinado time.
- '**`f nome_time`**': imprime a folha salarial consolidada de um determinado time.
- '**`g`**': imprime a folha salarial geral.
- '**`b`**': deve chamar a função '**avaliacao_basica()**' implementada no arquivo "**avaliacao_basica_nba.hpp**" (já incluído no main.cpp). Essa função faz uma avaliação do código (não apenas dos resultados).

Para ilustrar, abaixo são apresentados dois exemplos de entrada/saída:

```
input=
j LAClippers RobertCovington PF 12975471
j LAClippers ReggieJackson PG 10384500
j LAClippers MarcusMorrisSr. SF 15627907
j LAClippers NormanPowell SF 15517241
j LAClippers KawhiLeonard SF 39344900
j LAClippers PaulGeorge SG 39344900
j LAClippers LukeKennard SG 13347727
j LAClippers EricBledsoe SG LAClippers 18125000
l LAClippers
output=
LAClippers
KawhiLeonard SF 39344900
PaulGeorge SG 39344900
EricBledsoe SG 18125000
MarcusMorrisSr. SF 15627907
NormanPowell SF 15517241
LukeKennard SG 13347727
RobertCovington PF 12975471
ReggieJackson PG 10384500

input=
j GoldenStateWarriors AndrewWiggins SF 31579390
j GoldenStateWarriors StephenCurry PG 45780966
j BrooklynNets KevinDurant PF 42018900
j BrooklynNets JamesHarden SG 44310840
j LosAngelesLakers RussellWestbrook PG 44211146
j LosAngelesLakers LeBronJames SF 41180544
g
output=
BrooklynNets 86329740
GoldenStateWarriors 77360356
LosAngelesLakers 85391690
```

**Atenção:** Lembre-se de fazer a correta modularização utilizando os arquivos **.hpp** e **.cpp**.

**Dica 1:**
O código da avaliação básica pode ser copiado aqui, caso você queira depurar algo localmente.

**Dica 2:**
Você pode usar os códigos dos exercícios anteriores e o da avaliação básica para lhe ajudar a fazer toda a parte de entrada/saída.

**Referências:**
https://www.cplusplus.com/doc/tutorial/structures/
https://www.cplusplus.com/reference/stl/

## Arquivos requeridos

main.cpp

```cpp
// NÃO ALTERE ESSA LINHA
#include "avaliacao_basica_nba.hpp"

int main() {


    //
    // Adicione seu código aqui e faça as demais alterações necessárias
    //


    return 0;
}
```

Jogador.hpp

Jogador.cpp

Time.hpp

Time.cpp

RegistroNBA.hpp

RegistroNBA.cpp

VPL

◄ L01E05 - Fila atendimento (3,0 pts)

Seguir para...

L01E07 - Controle de versão (2,0 pts) ►