UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computação
Algoritmos e Estrutura de Dados

# Exercícios 07 :: STL

Instruções Gerais

- Implemente os exercícios em C++ (extensão cpp e compilador g++).
- Ao final, envie o arquivo main.cpp pelo Moodle.
- As bibliotecas necessárias são: **<iostream>**, **<vector>**, **<forward_list>**, **<stack>** e **<queue>**.

1) Escreva uma função que recebe o endereço de um array de inteiros e devolve um vector<int>, contendo os elementos do array na mesma ordem.

   Função: **vector<int> vet_to_vector(int v*, int n);**

   Referência de **vector**: http://www.cplusplus.com/reference/vector/vector/

2) Escreva uma função que recebe o endereço de um array de inteiros e devolve uma forward_list<int>, contendo os elementos do array na mesma ordem.

   Função: **forward_list<int> vet_to_flist(int* v, int n);**

   Referência de **forward_list**: https://cplusplus.com/reference/forward_list/forward_list/

3) Escreva uma função que recebe o endereço de um array de inteiros e devolve uma stack<int>, contendo os elementos do array na mesma ordem.

   Função: **stack<int> vet_to_stack(int v*, int n);**

   Referência de **stack**: http://www.cplusplus.com/reference/stack/stack/

4) Escreva uma função que recebe a referência de duas forward_list<int> e concatena seus conteúdos em um vector<int>, a ser devolvido pelo função.

   Função: **vector<int> list_concat(forward_list& list1, forward_list& list2);**

   Exemplo de uso:

```
forward_list<int> f1 = {1,2,3,4};
forward_list<int> f2 = {5,6};
vector<int> vec = list_concat(f1, f2);
// vec = {1,2,3,4,5,6};
```

5) Escreva uma função que verifica a parentização de expressões string infix do tipo “[(x + 8) * (9-2)]”. Utilize uma pilha stack<char> para realizar o processo:
   - percorrer a expressão de entrada
   - ao encontrar um símbolo "abre", ( [ { , empilha
   - ao encontrar um símbolo "fecha", ) ] }, retira o símbolo do tipo "abre" do topo e compara ambos
     - se o "abre" e o "fecha" correspondem, continua
     - se diferem, então a expressão está incorreta
     - se a pilha estiver vazia, a expressão está incorreta
   - ao terminar de percorrer a expressão de entrada, a pilha deve estar vazia
     - se restou algum símbolo na pilha, então a expressão está incorreta

   Função: **bool check_brackets(string expression);**

6) Escreva uma função que recebe uma expressão pós-fixada (notação Polonesa Inversa) como uma string e devolve um vector<string> contendo cada elemento da expressão. Note que existe um caractere de espaço entre cada elemento da expressão e que os números podem ser compostos por mais de um dígito. Utilize os recursos sugeridos da biblioteca padrão de C++.

   Função: **vector<string> vectorize_expression(string expression);**

```
Exemplo:   entrada:   "24 32 + 2 / 41 5 * +"
           saída:     vector<string> = {24, 32, +, 2, /, 41, 5, *, +}
```

   Recursos:
   - expressão posfix (Polonesa Inversa): https://en.wikipedia.org/wiki/Reverse_Polish_notation
     - considere operandos naturais e as quatro operações: * / + -

7) Escreva uma função que calcula e devolve o resultado de expressões posfix usando os recursos da biblioteca padrão de C/C++. Considere que sempre existe um espaço entre os elementos da expressão e que os números são inteiros positivos e podem ser formados por mais de um algarismo. Operações possíveis: + - * /. É necessário usar uma pilha para os operandos (stack<float>).

   Recursos:
   - isdigit(): verifica se caractere é um dígito (número)
     - http://www.cplusplus.com/reference/cctype/isdigit/
   - stoi(): converte string C++ para número (http://www.cplusplus.com/reference/string/stoi/)

   Processo:
   - Transforme a expressão string em um vector<string>. Desta forma, cada símbolo (operador ou operando) estará em uma posição do vector.
   - Varrer a expressão no vector:
     - Quando encontra um número na expressão, empilha (stack<float>).
     - Quando encontra um operador na expressão, retira dois elementos da pilha, executa a operação e empilha resultado (a ordem dos operandos deve ser considerada na execução da operação);
     - Quando terminar de percorrer a expressão, o resultado estará no topo da pilha

   Função: **float calc_posfix(string expression);**

8) Escreva uma função que verifica a corretude de uma expressão posfix. Para tanto, modifique a lógica da função do exercício anterior para que o processo ocorra da seguinte forma:
   - Quando encontra um número na expressão, empilha
   - Quando encontra um operador na expressão, deve haver ao menos dois números na pilha
     - retira apenas um número da pilha e prossegue na varredura da expressão
   - Quanto terminar de percorrer a expressão, deverá haver exatamente um número na pilha

   Função: **bool check_posfix(string expression);**

9) Escreva uma calculadora de expressões infix parentizada: ( ( ( 6 + 9 ) / 3 ) * ( 6 - 4) )
   - Considere que sempre existe um espaço entre os elementos da expressão e que ela está completamente parentizada para informar a ordem de precedência das operações.
   - São necessárias duas pilhas: operadores (stack <char>) e operandos (stack <float>)).
   - Utilize novamente os recursos da biblioteca C/C++ para facilitar a construção do programa.
   - Para avaliar a expressão deve-se seguir o processo abaixo:
     - Quando um operando (número) é lido, ele é colocado na pilha de operandos.
     - Quanto um parêntese esquerdo é lido "(", ele é colocado na pilha de operadores.
     - Quando um dos quatro operadores é lido, ele é colocado na pilha de operadores caso tenha maior precedência sobre o operador que está no topo da pilha ou topo seja “(“.
       - Caso contrário, deve calcular a expressão empilhada: retirar dois números da pilha de operandos e o operador da outra pilha. Calcular a expressão e colocar o resultado na pilha de operandos.
     - Quando um parêntese direito é lido ")", deve calcular a expressão empilhada seguindo o processo mencionado acima, colocando o resultado na pilha de operandos. Este processo é repetido até que o topo da pilha de operadores contenha um parêntese esquerdo "(", que deverá ser removido.

   Função: **float calc_infix(string expression);**

10) Escreva um conversor de expressões posfix para infix, usando uma versão modificada da calculadora posfix. Exemplo: **6 9 + 3 / 6 4 - *** → **( ( ( 6 + 9 ) / 3 ) * ( 6 - 4 ) )**
   - Utilize uma pilha de strings (stack<string>) para guardar as expressões na forma infix.
   - Utilize os recursos da biblioteca C/C++ para facilitar a construção do programa.
   - Para avaliar a expressão deve-se seguir o processo abaixo:
     - Quando um número (operando) é lido, ele é colocado na pilha.
     - Quando um operador é lido, deve montar a expressão infix com os dois operandos previamente empilhados: retira dois topos da pilha e monta expressão (string) na forma: **( topo operador topo )**. Empilha a expressão infix na pilha.
   - Ao final da expressão, a pilha deverá conter um único elemento, com a expressão completa na forma infix.

   Função: **string posfix_to_infix(string expression);**