

Tema: Introdução à programação VI
Atividade: Apontadores

INSTRUÇÕES:

- Desenvolver classes/métodos em C++ para atender às especificações abaixo.
- Providenciar a documentação essencial:
nome e matrícula,
identificação, objetivo, parâmetros e condições especiais,
se houver, e relatório de testes (exemplos de valores usados e condições testadas).

SUGESTÃO: Montar um menu para a escolha do método a ser testado
(ver modelo em Lista00.cpp).

Testes deverão ser realizados e os valores usados deverão
ser guardados no final do programa como comentários (`/* e */`).
O uso de recursão é opcional; se desejar utilizá-lo,
fazer também a implementação da forma não-recursiva.

0.) Editar programa em C++, na mesma pasta, cujo nome será Exemplo1700.cpp, para testar definições de métodos a serem desenvolvidos:

```
/*
    Exemplo1700 - v0.0. - __ / __ / ____
    Author: _____
*/

// ----- preparacao

// dependências

#include <iostream>

// ----- definicoes globais

using namespace std;

// ----- metodos

/**
    Method_00 - nao faz nada.
*/
void method_00 ( )
{
    // nao faz nada
} // end method_00 ( )

/**
    Method_01 - Testar definicoes da classe.
*/
void method_01 ( )
{
    // definir dados

    // identificar
    cout << "\nMethod_01 - v0.0\n" << endl;

    // encerrar
    pause ( "Apertar ENTER para continuar" );
} // end method_01 ( )
```

```

// ----- acao principal

/*
Funcao principal.
@return codigo de encerramento
*/
int main ( int argc, char** argv )
{
    // definir dado
    int x = 0;          // definir variavel com valor inicial

    // repetir até desejar parar
    do
    {
        // identificar
        cout << "EXEMPLO1700 - Programa - v0.0\n" << endl;

        // mostrar opcoes
        cout << "Opcoes" << endl;
        cout << " 0 - parar" << endl;
        cout << " 1 - testar definicoes" << endl;

        // ler do teclado
        cout << endl << "Entrar com uma opcao: ";
        cin >> x;

        // escolher acao
        switch ( x )
        {
            case 0:
                method_00 ( );
                break;
            case 1:
                method_01 ( );
                break;
            default:
                cout << endl << "ERRO: Valor invalido." << endl;
        } // end switch
    }
    while ( x != 0 );

    // encerrar
    pause ( "Apertar ENTER para terminar" );
    return ( 0 );
} // end main ( )

```

/*

----- documentacao complementar

----- notas / observacoes / comentarios

----- previsao de testes

----- historico

Versao	Data	Modificacao
0.1	__/__/__	esboco

----- testes

Versao	Teste	
0.1	01. (OK)	identificacao de programa

*/

Exercícios:

DICAS GERAIS: Consultar o Anexo CPP 02 na apostila para outros exemplos.

Prever, realizar e registrar todos os testes efetuados.

Desenvolver e testar cada um dos protótipos de métodos sugeridos abaixo, usando apenas apontadores.

Não usar métodos ou funções já prontos em bibliotecas nativas da linguagem.

Integrar as chamadas de todos os programas em um só.

Para os próximos exercícios considerar as seguintes definições de tipo/classe

```
typedef struct s_intStack { int length; int *data } intStack;
```

```
typedef intArray* ref_intStack;
```

01.)

```
/**
```

Funcao para acrescentar valor no topo de uma pilha (LIFO)
montada em um arranjo, por meio de apontador.

@return apontador para pilha atualizada

@param stack - apontador para pilha

@param value - valor a ser inserido

```
*/
```

```
ref_intStack intStack_push ( ref_intStack stack, int value )
```

02.)

```
/**
```

Funcao para remover valor do topo de uma pilha (LIFO)
montada em um arranjo, por meio de apontador.

@return apontador para pilha atualizada

@param stack - apontador para pilha

```
*/
```

```
ref_intStack intStack_pop ( ref_intStack stack )
```

03.)

```
/**
```

Funcao para duplicar valor no topo de uma pilha (LIFO)
montada em um arranjo, por meio de apontador.

@return apontador para pilha atualizada

@param stack - apontador para pilha

```
*/
```

```
ref_intStack intStack_dup ( ref_intStack stack )
```

04.)

```
/**  
    Funcao para trocar a ordem dos valores no topo de uma pilha (LIFO)  
    montada em um arranjo, por meio de apontador, se houver pelo menos dois valores.  
    @return apontador para pilha atualizada  
    @param stack - apontador para pilha  
*/  
ref_intStack intStack_swap ( ref_intStack stack )
```

05.)

```
/**  
    Funcao para inverter a ordem dos valores em uma pilha (LIFO)  
    montada em um arranjo, por meio de apontador, se houver pelo menos dois valores.  
    @return apontador para pilha atualizada  
    @param stack - apontador para pilha  
*/  
ref_intStack intStack_invert ( ref_intStack stack )
```

Para os próximos exercícios considerar as seguintes definições de tipo/classe

```
typedef struct s_intQueue { int length; int *data } intQueue;
```

```
typedef intArray* ref_intQueue;
```

06.)

```
/**  
    Funcao para acrescentar valor no final de uma fila (FIFO)  
    montada em um arranjo, por meio de apontador.  
    @return apontador para fila atualizada  
    @param queue - apontador para fila  
    @param value - valor a ser inserido  
*/  
ref_intQueue intQueue_push ( ref_intQueue queue, int value )
```

07.)

```
/**  
    Funcao para remover valor do início de uma fila (FIFO)  
    montada em um arranjo, por meio de apontador.  
    @return apontador para fila atualizada  
    @param queue - apontador para fila  
*/  
ref_intQueue intQueue_pop ( ref_intQueue queue )
```

08.)

/**

Funcao para comparar filas de inteiros
por meio de apontadores.

@return zero , se forem iguais;

negativo, se o valor da diferenca for menor e estiver na primeira fila

positivo , se o valor da diferenca for maior e estiver na primeira fila

@param p - apontador para inicio da primeira fila

@param q - apontador para inicio da segunda fila

*/

ref_intQueue intQueue_compare (ref_intQueue p, ref_intQueue q)

09.)

/**

Funcao para juntar filas de inteiros
por meio de apontadores.

@return fila resultante da fusão

@param p - apontador para inicio da primeira fila

@param q - apontador para inicio da segunda fila

*/

ref_intQueue intQueue_join (ref_intQueue p, ref_intQueue q)

10.)

/**

Funcao para procurar valor em uma fila (FIFO)
montada em um arranjo, por meio de apontador.

@return 1, se encontrar; 0, caso contrário

@param queue - apontador para fila

@param value - valor a ser procurado

*/

int intQueue_search (ref_intQueue queue, int value)

Tarefas extras

E1.)

/**

Funcao para intercalar filas de inteiros
por meio de apontadores.

@return apontador para fila resultante da fusão

@param p - apontador para inicio da primeira fila

@param q - apontador para inicio da segunda fila

*/

ref_intQueue intQueue_merge (ref_intQueue p, ref_intQueue q)

E2.)

/**

Funcao para intercalar filas de inteiros em ordem crescente
por meio de apontadores. Descartar valores iguais.

@return apontador para fila ordenada e filtrada

@param p - apontador para inicio da primeira fila

@param q - apontador para inicio da segunda fila

*/

ref_intQueue intQueue_mergeUp (ref_intQueue p, ref_intQueue q)