

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 Exemplo1600.cpp, para testar definições de métodos a serem desenvolvidos:

```
/*
    Exemplo1600 - 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 << "EXEMPLO1600 - 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.

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

Prever, realizar e registrar todos os testes efetuados.

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

Integrar as chamadas de todos os testes em um só programa.

Os métodos deverão para C++ e buscar compatibilidade com a linguagem C.

Restrições:

- Usar definições de dados e métodos em classes, como a para arranjos definida anteriormente, adaptada para usar *template*, de preferência.
- Não usar *break*!

01.)

```
/**  
    Funcao para acrescentar valor ao final  
    de um arranjo, por meio de apontador.  
    @return  apontador para arranjo atualizado  
    @param array - apontador para arranjo  
    @param value - valor a ser inserido  
*/  
int* array_push_back ( int *array, int value )
```

02.)

```
/**  
    Funcao para remover valor do final  
    de um arranjo, por meio de apontador.  
    @return  apontador para arranjo atualizado, ou NULL,  
    @param array - apontador para arranjo  
*/  
int* array_pop_back ( int *array )
```

03.)

```
/**  
    Funcao para acrescentar valor ao início  
    de um arranjo, por meio de apontador.  
    @return  apontador para arranjo atualizado  
    @param array - apontador para arranjo  
    @param value - valor a ser inserido  
*/  
int* array_push_front ( int value, int *array )
```

04.)

```
/**  
    Funcao para remover valor do início  
    de um arranjo, por meio de apontador.  
    @return apontador para arranjo atualizado, ou NULL,  
    @param array - apontador para arranjo  
*/  
int* array_pop_front ( int *array )
```

05.)

```
/**  
    Funcao para acrescentar valor no meio (aproximadamente)  
    de um arranjo, por meio de apontador.  
    @return apontador para arranjo atualizado  
    @param array - apontador para arranjo  
    @param value - valor a ser inserido  
*/  
int* array_push_mid ( int *array, int value )
```

06.)

```
/**  
    Funcao para remover valor do meio (aproximadamente)  
    de um arranjo, por meio de apontador.  
    @return apontador para arranjo atualizado, ou NULL,  
    @param array - apontador para arranjo  
*/  
int* array_pop_mid ( int *array )
```

07.)

```
/**  
    Funcao para comparar arranjos de inteiros  
    por meio de apontadores.  
    @return zero , se forem iguais;  
            negativo, se o valor da diferenca for menor e estiver no primeiro arranjo  
            positivo , se o valor da diferenca for maior e estiver no primeiro arranjo  
    @param p - apontador para inicio do primeiro arranjo  
    @param q - apontador para inicio do segundo arranjo  
*/  
int intArray_cmp ( ref_intArray p, ref_intArray q )
```

08.)

```
/**  
    Funcao para juntar um arranjo de inteiros ao final de outro  
    por meio de apontadores.  
    @return apontador para inicio do arranjo com a uniao  
    @param p - apontador para inicio do primeiro arranjo  
    @param q - apontador para inicio do segundo arranjo  
    */  
ref_intArray intArray_append ( ref_intArray p, ref_intArray q )
```

09.)

```
/**  
    Funcao para procurar pela primeira ocorrencia de valor em arranjo  
    por meio de apontador.  
    @return apontador para a primeira ocorrência; NULL, caso contrario  
    @param a - apontador para arranjo de inteiros  
    @param x - valor ser procurado  
    */  
ref_intArray intArray_seek ( ref_intArray a, int x )
```

10.)

```
/**  
    Funcao para separar sequencia de valores em arranjo  
    por meio de apontador.  
    @return apontador para inicio da sequencia de inteiros; NULL, caso contrario  
    @param a - apontador para arranjo de inteiros  
    @param start - posicao inicial  
    @param size - quantidade de dados  
    */  
ref_intArray intArray_sub ( ref_intArray a, int start, int size )
```

## Tarefas extras

E1.)

```
/**  
    Funcao para intercalar arranjos de inteiros, usando um valor de cada por vez,  
    por meio de apontadores.  
    @return apontador para inicio do arranjo com a fusao  
    @param p - apontador para inicio do primeiro arranjo  
    @param q - apontador para inicio do segundo arranjo  
*/  
ref_intArray intArray_merge ( ref_intArray p, ref_intArray q )
```

E2.)

```
/**  
    Funcao para mesclar arranjos de inteiros em ordem de decrescente  
    por meio de apontadores.  
    @return apontador para inicio do arranjo com a fusao  
    @param p - apontador para inicio do primeiro arranjo  
    @param q - apontador para inicio do segundo arranjo  
*/  
ref_intArray intArray_mergeDown ( ref_intArray p, ref_intArray q )
```