Aula 9

Objectivo: Revisões;
Introdução aos templates.

Exercício 1

Adapte o código da lista ligada implementada na aula 5 de forma a que a classe CLinkedList seja agora uma classe template. O tipo de dados armazenado na lista deixa de ser void* e passa a ser um tipo de dados genérico T.
Note que a implementação de uma classe template, ao contrário das outras classes, é feita no ficheiro *.h. O código apresentado abaixo corresponde ao ficheiro cabeçalho da sua nova lista e já fornece algumas funções. Complete-o obedecendo aos comentários inseridos no corpo das restantes funções.

```
template <class T> class CLinkedList
{
protected:
	struct SNode
	{
		T m_data;
		SNode *m_pNext;
		int m_key;
	};

public:
	CLinkedList()
	{
		m_pHead = m_pCurrent = NULL;
	}

	virtual ~CLinkedList()
	{
		Empty();
	}

public:
	int InsertHead(const T& data, const int key)
	{
		SNode *pNode;

		pNode = new SNode;
		if (pNode == NULL)
			return OUT_OF_MEMORY;

		pNode->m_data = data;
		pNode->m_pNext = m_pHead;
		pNode->m_key = key;
		m_pHead = pNode;
		++m_count;
		return OK;
	}

	void Empty()
	{
		SNode *pNode = m_pHead;
```

```
            while (pNode != NULL)
            {
                  m_pHead = pNode->m_pNext;
                  delete pNode;
                  pNode = m_pHead;
            }

            m_count = 0;
            m_pCurrent = NULL;
      }

      bool IsEmpty()
      {
            return (m_pHead == NULL);
      }

      unsigned int Count()
      {
            return m_count;
      }

      int InsertTail(const T& data, const int key)
      {//insere um elemento na última posição da lista}

      int RemoveHead(T& data)
      {//remove o primeiro elemento da lista}

      int Remove(int key)
      {//remove o elemento identificado pela chave}

      T* GetHead()
      {//devolve um ponteiro para os dados do primeiro elemento da lista}

      T* GetHead(int& key)
      {//devolve um ponteiro para os dados do primeiro elemento da lista
       //e a respectiva chave}

      T* GetNext()
      {//devolve um ponteiro para os dados do elemento actual da lista}

      T* GetNext(int& key)
      {//devolve um ponteiro para os dados do elemento actual da lista
       //e a respectiva chave}

      T* GetData(int key)
      {//devolve um ponteiro para os dados do elemento identificado pela chave}

protected:
      unsigned int m_count;
      SNode *m_pHead, *m_pCurrent;
};
```

Exercício 2

Para resolver este exercício utilize os ficheiros fonte disponíveis no sítio da disciplina: book.h, book.cpp, magazine.h, magazine.cpp, libitem.h e libitem.cpp. Adapte a classe CLibrary que implementou na aula 3 de forma a que esta utilize a lista ligada do exercício anterior.

```
class CLibrary
{
public:
      CLibrary();
      ~CLibrary();
```

```
public:
	int CreateBook(const char* pTitle, unsigned int year,
			const char* pAuthor, unsigned int numPages);
	int CreateMagazine(const char* pTitle, unsigned int year,
				EPeriodicity periodicity, unsigned int number);
	int Delete(int code);
	int Print(int code);
	void PrintAll();
	void Search(const char* pTitle);

private:
	CLinkedList<CLibItem*> m_libList;
	static int m_code;
};
```

Teste esta classe construindo um programa que, por intermédio de um menu, lhe permita:

```
C - Adicionar livro
R - Adicionar revista
A - Apagar um item identificado pelo código
V - Visualizar um item identificado pelo código
L - Listar todos os items
X - Sair do programa
```