

== Laboratório 6 ==

1. Escreva uma classe `Carrinho` para armazenar `Itens` de supermercado, definidos a partir da classe `Item` e de suas classes derivadas. A classe `Carrinho` define como atributo privado um ponteiro para ponteiros para objetos da classe `Item` (`Item **ptr`). Esse ponteiro é usado para armazenar o endereço de um array de ponteiros para para objetos da classe `Item`. Além disso, a classe `Carrinho` ainda define como atributos privados um inteiro para definir o tamanho máximo do array e outro para controlar o índice do elemento no topo do array. Todos os atributos são inicializados no construtor, sendo que o array é iniciado dinamicamente.

A classe `Carrinho` implementa ainda um destrutor e um método de inserção e de exibição. O método de inserção é realizado através da sobrecarga do operador `()`, enquanto a exibição é implementada a partir da sobrecarga de `[]`. No carrinho, são inseridos objetos da classe `Item`, que possuem um método `print` virtual para a impressão dos valores associados aos atributos privados: `nome` e `preço`. Ainda, são inseridos no carrinho objetos de uma classe derivada da classe `Item`, chamada `ItemCarnaval`, que possui como atributo adicional o nome do supermercado onde o item pode ser encontrado. Tanto a classe `Item` quanto a classe `ItemCarnaval` não possuem construtores padrão. Tendo em vista a função principal abaixo e a saída na tela, escreva o código contido nos arquivos `carrinho.h` e `.cpp`, `item.h` e `.cpp` e `itemcarnaval.h` e `.cpp`. Saiba que o operador `<<` foi sobrecarregado apenas na classe `Item`.

```

/*****
/***** Programa Principal *****/

int main() {
    Item arroz ("Arroz", 10.00);
    Item feijao ("Feijao", 8.00);

    Carrinho c (5);
    c(arroz);
    c(feijao);
    cout << c[0] << endl << c[1] << endl;

    ItemCarnaval cerveja ("Heineken", 2.50, "Extra");
    ItemCarnaval macarrao ("Miojo", 4.00, "Sendas");

    c (cerveja);
    c (macarrao);
    cout << c[2] << endl << c[3] << endl;

    return 0;
}

/*****
/***** Saída na tela *****/

Arroz      10
Feijão     8
Heineken   2,5      Extra
Miojo      4        Sendas

/*****/
```

2. Reescreva o programa da Questão 1, substituindo o array de ponteiros para objetos da classe `Item` por um vector STL de ponteiros para objetos da classe

Item. Note que apenas os arquivos `carrinho.h` e `carrinho.cpp`, precisam ser atualizados.

3. Reescreva o programa da Questão 1 para que uma exceção do tipo `CarrinhoCheioException` seja disparada caso um produto seja inserido com o carrinho já cheio. Note que a quantidade de produtos no carrinho é regulada a partir do atributo `tamanho`. Ainda, note que a classe `CarrinhoCheioException` dispara uma mensagem de aviso sobre a exceção ocorrida e que ela é definida a partir de uma classe derivada da classe `exception`.

Reescreva os arquivos `carrinho.h` e `carrinho.cpp` e crie os arquivos `carrinhocheioexception.h` e `carrinhocheioexception.cpp`.

4. Reescreva novamente o programa da Questão 1, tornando a classe `Carrinho` uma classe derivada da classe STL `map`. Nesse caso, o operador sobrecarregado `[]` utilizado na função principal deixa de acessar os elementos no carrinho a partir do índice e passa a acessá-los pelo nome. Note que tanto o operador `[]` quanto o operador `()` devem ser alterados. Ainda, o atributo `topo` da classe `Carrinho` deixa de ser útil já que a estrutura não é mais seqüencial. Por fim, para inserir o elemento no carrinho, é necessário criar um método `getNome()` na classe `Item`.

Reescreva a função principal e os arquivos `carrinho.h`, `carrinho.cpp`, `item.h` e `item.cpp`

## == Respostas ==

### 1.

```

/*****
/***** carrinho.h *****/

#include <iostream>
#include "item.h"

#ifndef CARRINHO_H
#define CARRINHO_H

using namespace std;

class Carrinho {
public:
    Carrinho (int = 10);
    ~Carrinho ();

    void operator() (Item &);

    Item &operator[] (int);

private:
    int tamanho, topo;
    Item **lista;
};

#endif

/*****
/***** carrinho.cpp *****/

#include "carrinho.h"

Carrinho::Carrinho (int t) : tamanho (t), topo (-1), lista (new Item *[t]) {}

Carrinho::~Carrinho () { delete [] lista; }
```

```

void Carrinho::operator() (Item &item) {
    lista [++topo] = &item;
}

Item &Carrinho::operator[] (int idx) {
    return *lista [idx];
}

/*****
/***** item.h *****/

#include <iostream>

#ifndef ITEM_H
#define ITEM_H

using namespace std;

class Item {
    friend ostream &operator<< (ostream &, Item &);

public:
    Item (string, double);
    virtual void print ();

private:
    string nome;
    double preco;
};

#endif

/*****
/***** item.cpp *****/

#include "item.h"

ostream &operator<< (ostream &out, Item &i) {
    i.print ();
    return out;
}

Item::Item (string n, double p): nome (n), preco (p) {}

void Item::print () {
    cout << nome << "\t" << preco;
}

/*****
/***** itemcarnaval.h *****/

#include <iostream>
#include "item.h"

#ifndef ITEMCARNAVAL_H
#define ITEMCARNAVAL_H

using namespace std;

class ItemCarnaval : public Item {
public:
    ItemCarnaval (string, double, string);
    virtual void print ();

private:
    string supermercado;
};

#endif

/*****
/***** itemcarnaval.cpp *****/

#include "itemcarnaval.h"

ItemCarnaval::ItemCarnaval (string n, double p, string s) :

```

```

Item (n,p), supermercado (s) {}

void ItemCarnaval::print () {
    Item::print ();
    cout << "\t" << supermercado;
}

/*****/

```

## 2.

```

/*****/
/***** carrinho.h *****/

#include <iostream>
#include <vector>
#include "item.h"

#ifndef CARRINHO_H
#define CARRINHO_H

using namespace std;

class Carrinho {
public:
    Carrinho (int = 10);

    void operator()(Item &);

    Item &operator[](int);

private:
    int tamanho, topo;
    vector <Item *> vetor;
};

#endif

/*****/
/***** carrinho.cpp *****/

#include "carrinho.h"

Carrinho::Carrinho (int t): tamanho (t), topo (-1), vetor (t) {}

void Carrinho::operator()(Item &item) {
    vetor [++topo] = &item;
}

Item &Carrinho::operator[](int idx) {
    return *vetor [idx];
}

/*****/

```

## 3.

```

/*****/
/***** Programa Principal *****/

#include <iostream>
#include <string>
#include <vector>
#include "carrinho.h"
#include "item.h"
#include "itemcarnaval.h"

using namespace std;

int main() {
    Item arroz ("Arroz", 10.00);
    Item feijao ("Feijao", 8.00);

    Carrinho c (5);
    c(arroz);
}

```

```

    c(feijao);
    cout << c[0] << endl << c[1] << endl;

    ItemCarnaval cerveja ("Heineken", 2.50, "Extra");
    ItemCarnaval macarrao ("Miojo", 4.00, "Sendas");

    c (cerveja);
    c (macarrao);
    cout << c[2] << endl << c[3] << endl;

    ItemCarnaval limao ("Limao", 0.50, "Prezunic");
    c (limao);

    return 0;
}

/*****
/***** Arquivo carrinho.h *****/

#include <iostream>
#include "item.h"
#include "carrinhocheioexception.h"

#ifndef CARRINHO_H
#define CARRINHO_H

using namespace std;

class Carrinho {
public:
    Carrinho (int = 10);
    ~Carrinho ();

    void operator() (Item &);

    Item &operator[] (int);

private:
    int tamanho, topo;
    Item **lista;
};

#endif

/*****
/***** Arquivo carrinho.cpp *****/

#include "carrinho.h"

Carrinho::Carrinho (int t) : tamanho (t), topo (-1), lista (new Item *[t]) {}

Carrinho::~Carrinho () { delete [] lista; }

void Carrinho::operator() (Item &item) {
    try {
        if (topo == tamanho - 1) throw CarrinhoCheioException ();
        lista [++topo] = &item;
    } catch (CarrinhoCheioException &e) {
        cout << "Falha na insercao: " << e.what () << endl;
    }
}

Item &Carrinho::operator[] (int idx) {
    return *lista [idx];
}

/*****
/***** Arquivo carrinhocheioexception.h *****/

#include <exception>

using namespace std;

#ifndef CARRINHOICHEIEXCEPTION_H
#define CARRINHOICHEIEXCEPTION_H

class CarrinhoCheioException : public exception {

```

```

        public:
            virtual const char * what () const throw ();
};

#endif

/*****
/***** Arquivo carrinhocheioexception.cpp *****/

#include "carrinhocheioexception.h"

const char *CarrinhoCheioException::what () const throw () {
    return "Carrinho cheio!";
}

/*****

```

#### 4.

```

/*****
/***** Programa Principal *****/

#include <iostream>
#include <string>
#include <vector>
#include "carrinho.h"
#include "item.h"
#include "itemcarnaval.h"

using namespace std;

int main() {
    Item arroz ("Arroz", 10.00);
    Item feijao ("Feijao", 8.00);

    Carrinho c (5);
    c(arroz);
    c(feijao);
    cout << c["Arroz"] << endl << c["Feijao"] << endl;

    ItemCarnaval cerveja ("Heineken", 2.50, "Extra");
    ItemCarnaval macarrao ("Miojo", 4.00, "Sendas");

    c (cerveja);
    c (macarrao);
    cout << c["Heineken"] << endl << c["Miojo"] << endl;

    return 0;
}

/*****
/***** Arquivo carrinho.h *****/

#include <iostream>
#include <map>
#include "item.h"

#ifndef CARRINHO_H
#define CARRINHO_H

using namespace std;

typedef map<string, Item *> meuMap;

class Carrinho : public meuMap {
public:
    Carrinho (int = 10);

    void operator()(Item &);

    Item &operator[](string);

private:
    int tamanho, topo;
};

```

```

#endif

/*****
/***** Arquivo carrinho.cpp *****/

#include "carrinho.h"

Carrinho::Carrinho (int t):
    tamanho (t) {}

void Carrinho::operator() (Item &item) {
    insert (meuMap::value_type (item.getNome (), &item));
}

Item &Carrinho::operator[] (string nome) {
    return *(find (nome)->second);
}

/*****
/***** Arquivo item.h *****/

#include <iostream>

#ifndef ITEM_H
#define ITEM_H

using namespace std;

class Item {
    friend ostream &operator<< (ostream &, Item &);

public:
    Item (string, double);
    string getNome () const;

    virtual void print ();

private:
    string nome;
    double preco;
};

#endif

/*****
/***** Arquivo item.cpp *****/

#include "item.h"

ostream &operator<< (ostream &out, Item &i) {
    i.print ();
    return out;
}

Item::Item (string n, double p): nome (n), preco (p) {}

string Item::getNome () const { return nome; }

void Item::print () {
    cout << nome << "\t" << preco;
}

/*****

```