Tipos Abstratos de Dados (ADT)

Os Tipos Abstratos de Dados também denominados Abstract Data Types (ADT), consistem em uma forma de definir um novo tipo de dado juntamente com as operações que manipulam este novo tipo.

Utilizando orientação a objetos podemos criar novos Tipos Abstratos de Dados.

Implementação Sacola

In [1]:
```
#include <iostream>
using namespace std;

#define MAX 5
```

Definição da Classe

In [2]:
```
class Sacola {
    private:
        int capacidade;
        int dados[MAX];
        int contador;
    public:
        Sacola();
        bool vazia();
        bool cheia();
        bool inserir(int x);
        bool ocorrencia(int x);
        bool remover(int x);
};
```

Método Construtor

In [3]:
```
Sacola::Sacola() {
    contador = 0;
    capacidade = MAX;
}
```

In [4]:
```
Sacola minhaSacola;
```

Verifica se sacola está vazia

In [5]:
```
bool Sacola::vazia() {
    if (contador == 0)
        return true;
    else
        return false;
}
```

In [6]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!";
}
```

```
Está vazia!
```

Verifica se sacola está cheia

In [7]:
```
bool Sacola::cheia() {
    if (contador == capacidade)
        return true;
    else
        return false;
}
```

In [8]:
```
if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
}
```

```
Não está cheia!
```

Inserção elemento na sacola

In [9]:
```
bool Sacola::inserir(int x) {
    if (cheia())
        return false;
    dados[contador] = x;
    contador ++;
    return true;
}
```

In [10]:
```
if (minhaSacola.inserir(1))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!";
```

```
Inserção ok!
```

In [11]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}
```

```
Não está vazia!
```

In [12]:
```
if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Não está cheia!
```

In [13]:
```
if (minhaSacola.inserir(2))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!";
```

```
Inserção ok!
```

In [14]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}
```

```
Não está vazia!
```

In [15]:
```
if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Não está cheia!
```

In [16]:
```
if (minhaSacola.inserir(3))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!";
```

```
Inserção ok!
```

In [17]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}
```

```
Não está vazia!
```

In [18]:
```
if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Não está cheia!
```

In [19]:
```
if (minhaSacola.inserir(3))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!";
```

```
Inserção ok!
```

In [20]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}
```

```
Não está vazia!
```

In [21]:
```
if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Não está cheia!
```

In [22]:
```
if (minhaSacola.inserir(5))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!"
```

```
Inserção ok!
```

In [23]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}
```

```
Não está vazia!
```

In [24]:
```
if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Está cheia!
```

In [25]:
```
if (minhaSacola.inserir(4))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!"
```

```
Não iseriu!
```

In [26]:
```
if (minhaSacola.inserir(5))
    cout << "Inserção ok!";
else
    cout << "Não iseriu!"
```

```
Não iseriu!
```

Verifica ocorrência de elemento

In [27]:
```
bool Sacola::ocorrencia(int x) {
    for (int i = 0 ;  i < contador ; i ++)
        if (dados[i] == x)
            return true;
    return false;
}
```

In [28]:
```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}

if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Não está vazia!
Está cheia!
```

In [29]:
```
if (minhaSacola.ocorrencia(4))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Não existe!
```

In [30]:
```
if (minhaSacola.ocorrencia(5))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Existe!
```

In [31]:
```
if (minhaSacola.ocorrencia(6))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Não existe!
```

In [32]:
```
if (minhaSacola.ocorrencia(1))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Existe!
```

In [33]:
```
if (minhaSacola.ocorrencia(2))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Existe!
```

In [34]:
```
if (minhaSacola.ocorrencia(3))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Existe!
```

In [35]:
```
if (minhaSacola.ocorrencia(4))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Não existe!
```

In [36]:
```
if (minhaSacola.ocorrencia(5))
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Existe!
```

In [37]:
```
if (minhaSacola.ocorrencia(6))
```

```
    cout << "Existe!";
else
    cout << "Não existe!";
```

```
Não existe!
```

In [38]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Existe!
2: Existe!
3: Existe!
4: Não existe!
5: Existe!
6: Não existe!
```

Remoção de elemento da sacola

In [39]:

```
bool Sacola::remover(int x) {
    if (vazia())
        return false;
    for (int i = 0 ;  i < contador ; i ++) {
        if (dados[i] == x) {
            contador--;
            for (int j = i ; j < contador ; j ++) {
                dados[j] = dados[j+1];
            }
            return true;
        }
    }
    return false;
}
```

In [40]:

```
if (minhaSacola.remover(4))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Não removeu
```

In [41]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Existe!
2: Existe!
3: Existe!
4: Não existe!
5: Existe!
6: Não existe!
```

In [42]:

```
if (minhaSacola.remover(2))
    cout << "Remoção ok";
```

```
else
    cout << "Não removeu";
```

```
Remoção ok
```

In [43]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Existe!
2: Não existe!
3: Existe!
4: Não existe!
5: Existe!
6: Não existe!
```

In [44]:

```
if (minhaSacola.remover(5))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Remoção ok
```

In [45]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Existe!
2: Não existe!
3: Existe!
4: Não existe!
5: Não existe!
6: Não existe!
```

In [46]:

```
if (minhaSacola.remover(3))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Remoção ok
```

In [47]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Existe!
2: Não existe!
3: Existe!
4: Não existe!
5: Não existe!
6: Não existe!
```

In [48]:

```
if (minhaSacola.remover(3))
```

```
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Remoção ok
```

In [49]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Existe!
2: Não existe!
3: Não existe!
4: Não existe!
5: Não existe!
6: Não existe!
```

In [50]:

```
if (minhaSacola.remover(3))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Não removeu
```

In [51]:

```
if (minhaSacola.remover(1))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Remoção ok
```

In [52]:

```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Não existe!
2: Não existe!
3: Não existe!
4: Não existe!
5: Não existe!
6: Não existe!
```

In [53]:

```
if (minhaSacola.vazia()) {
    cout << "Está vazia!\n";
} else {
    cout << "Não está vazia!\n";
}

if (! minhaSacola.cheia()) {
    cout << "Não está cheia!";
} else {
    cout << "Está cheia!";
}
```

```
Está vazia!
Não está cheia!
```

In [54]:
```
if (minhaSacola.remover(2))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Não removeu
```

In [55]:
```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Não existe!
2: Não existe!
3: Não existe!
4: Não existe!
5: Não existe!
6: Não existe!
```

In [56]:
```
if (minhaSacola.remover(1))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Não removeu
```

In [57]:
```
if (minhaSacola.inserir(2))
    cout << "Remoção ok";
else
    cout << "Não removeu";
```

```
Remoção ok
```

In [58]:
```
for (int i = 0 ; i < 7 ; i ++)
    if (minhaSacola.ocorrencia(i))
        cout << i << ": Existe!\n";
    else
        cout << i << ": Não existe!\n";
```

```
0: Não existe!
1: Não existe!
2: Existe!
3: Não existe!
4: Não existe!
5: Não existe!
6: Não existe!
```

Exemplo 01:

In [59]:
```
%%file exemplo01/Sacola.h
#define MAX 5

class Sacola {
    private:
        int capacidade;
        int dados[MAX];
        int contador;
    public:
        Sacola();
        bool vazia();
```

```
        bool cheia();
        bool inserir(int x);
        bool ocorrencia(int x);
        bool remover(int x);
};
```

```
Overwriting exemplo01/Sacola.h
```

In [60]:

```
%%file exemplo01/Sacola.cpp
#include <cstdlib>
#include "Sacola.h"

Sacola::Sacola() {
    contador = 0;
    capacidade = MAX;
}

bool Sacola::vazia() {
    if (contador == 0)
        return true;
    else
        return false;
}

bool Sacola::cheia() {
    if (contador == capacidade)
        return true;
    else
        return false;
}

bool Sacola::inserir(int x) {
    if (cheia())
        return false;
    dados[contador] = x;
    contador ++;
    return true;
}

bool Sacola::ocorrencia(int x) {
    for (int i = 0 ;  i < contador ; i ++)
        if (dados[i] == x)
            return true;
    return false;
}

bool Sacola::remover(int x) {
    if (vazia())
        return false;
    for (int i = 0 ;  i < contador ; i ++) {
        if (dados[i] == x) {
            contador--;
            for (int j = i ; j < contador ; j ++) {
                dados[j] = dados[j+1];
            }
            return true;
        }
    }
    return false;
}
```

```
Overwriting exemplo01/Sacola.cpp
```

In [61]:

```
%%file exemplo01/teste01.cpp
```

```
#include <iostream>
#include "Sacola.h"

using namespace std;

int main() {
    Sacola minhaSacola;

    if (minhaSacola.vazia()) {
        cout << "Está vazia!";
    }

}
```

```
Overwriting exemplo01/teste01.cpp
```

In [62]:
```
!g++ exemplo01/teste01.cpp exemplo01/Sacola.cpp -o exemplo01/teste01
```

In [63]:
```
!./exemplo01/teste01
```

```
Está vazia!
```

Exemplo 02:

In [64]:
```
%%file exemplo01/teste02.cpp
#include <iostream>
#include <locale.h>
#include "Sacola.cpp"

using namespace std;


int main() {
    setlocale(LC_ALL, "portuguese");

    Sacola minhaSacola;

    if (minhaSacola.vazia()) {
        cout << "Está vazia!";
    }

}
```

```
Overwriting exemplo01/teste02.cpp
```

In [65]:
```
!g++ exemplo01/teste02.cpp -o exemplo01/teste02
```

In [66]:
```
!./exemplo01/teste02
```

```
Está vazia!
```

Se a implementação for modificada, os programas que utilizam o ADT não terão que ser alterados

Referências:

Presentation copyright 1995, The Benjamin/Cummings Publishing Company, For use with Data Structures and Other Objects by Michael Main and Walter Savitch.

Translation to portuguese by Prof. Maria Carolina Monard, ICMC-USP.

Modifications for C++ language by Prof. José Augusto Baranauskas, FFCLRP-USP, 2005

Update and modifications for use in Jupyter by Prof. Mateus Tarcinalli Machado, FATEC - Ribeirão Preto, 2020