

MFO

César Eduardo de Souza, Guilherme Diel

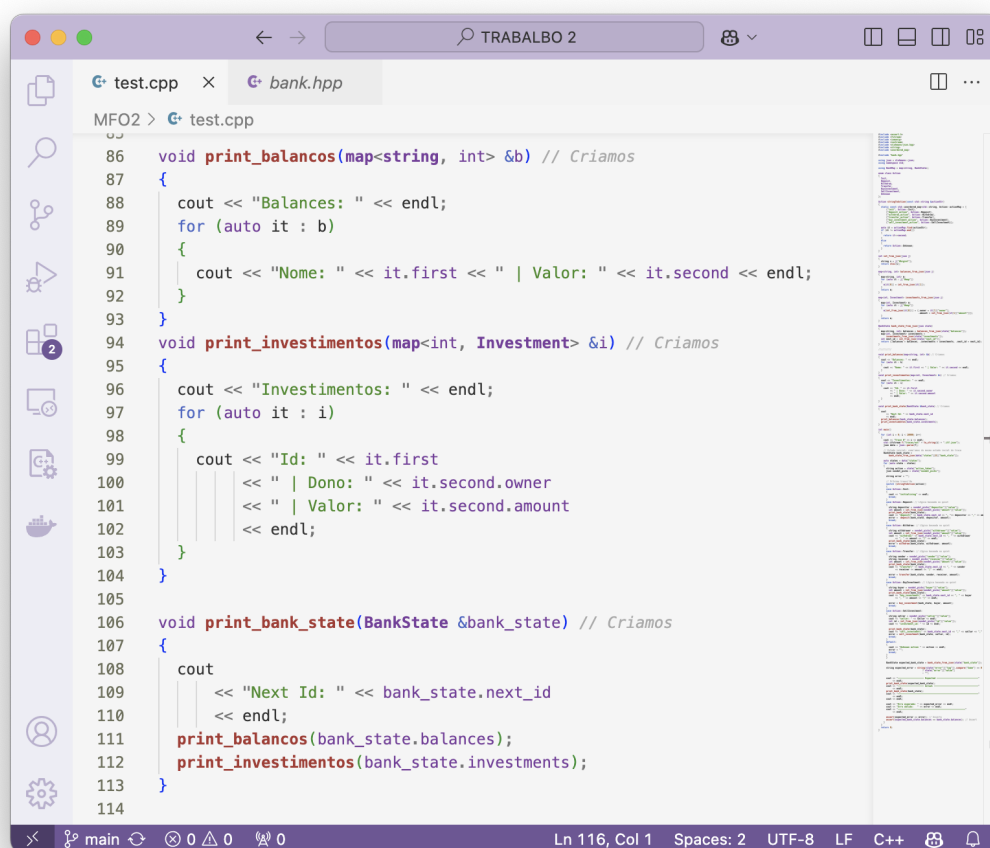
1. Descrição do teste

Alicerçou-se fortemente em alterações ao arquivo teste.cpp, pois qualificou-se o tal como um ambiente de trabalho provido de maior versatilidade, em comparação ao bank.hpp. Todavia, o segundo recebeu também mudanças em algumas definições, subsequentemente dispostas no presente documento.

2. Descrição das modificações

Primeiramente atualizou-se o software *quint*, em seguida, instalando-se a extensão para arquivos *.json* do *Visual Studio Code*.

Portanto, como é possível atestar em vista da codificação, foram desenvolvidas 3 novas funções para auxílio na execução. As tais apresentam-se na figura subsequente.



```
86 void print_balancos(map<string, int> &b) // Criamos
87 {
88     cout << "Balances: " << endl;
89     for (auto it : b)
90     {
91         cout << "Nome: " << it.first << " | Valor: " << it.second << endl;
92     }
93 }
94 void print_investimentos(map<int, Investment> &i) // Criamos
95 {
96     cout << "Investimentos: " << endl;
97     for (auto it : i)
98     {
99         cout << "Id: " << it.first
100            << " | Dono: " << it.second.owner
101            << " | Valor: " << it.second.amount
102            << endl;
103     }
104 }
105
106 void print_bank_state(BankState &bank_state) // Criamos
107 {
108     cout
109         << "Next Id: " << bank_state.next_id
110         << endl;
111     print_balancos(bank_state.balances);
112     print_investimentos(bank_state.investments);
113 }
114
```

Figura 1. Funções C++ de autoria própria.

Destarte, o autor inseriu dramáticas modificações à função

```
int main();
```

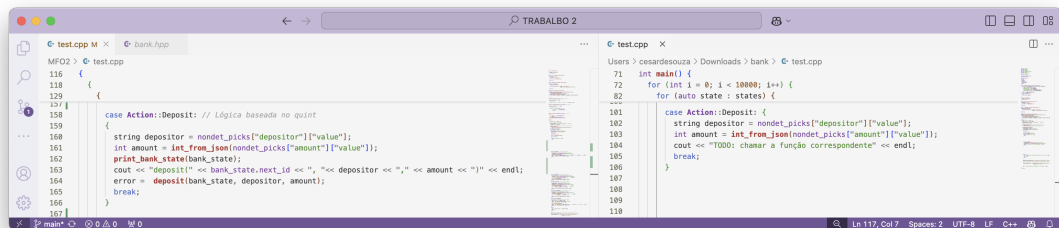
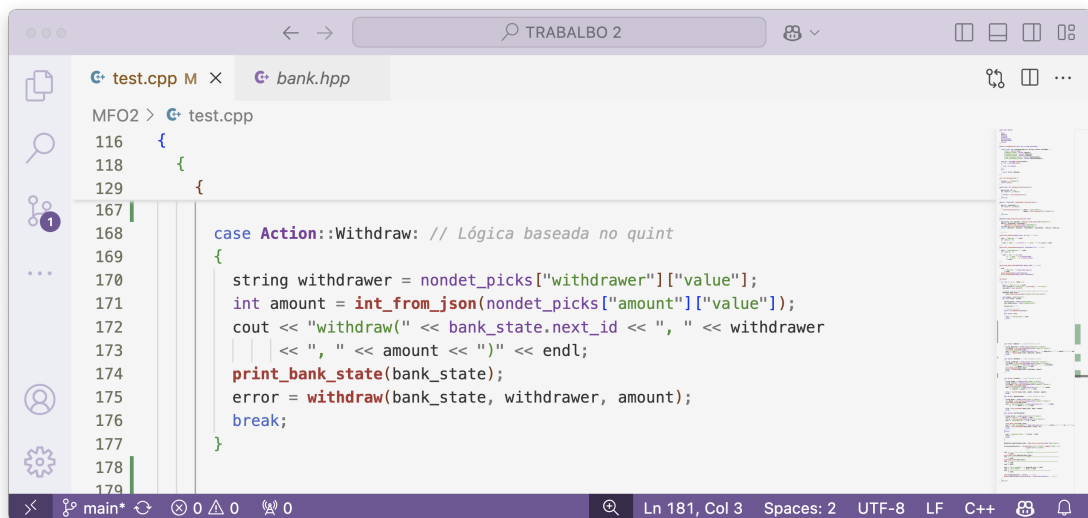
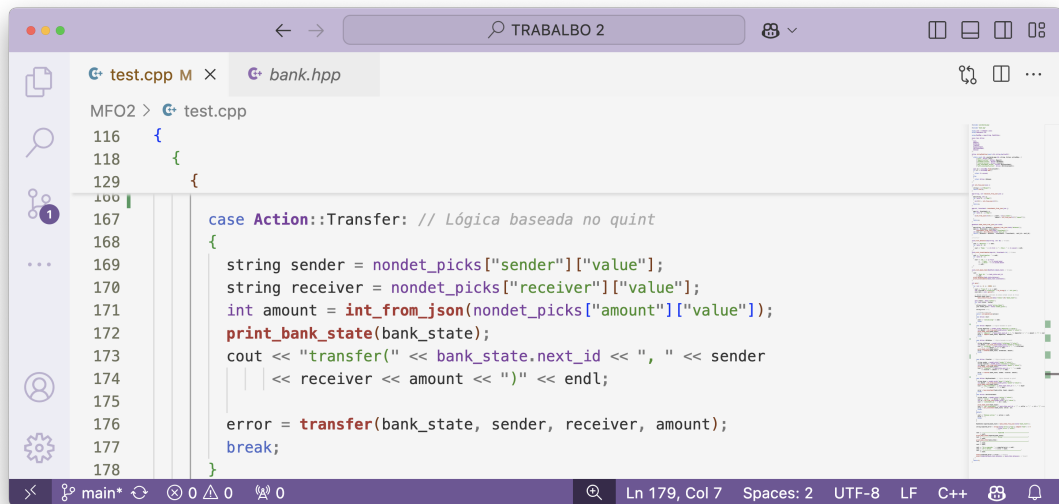


Figura 2. Case Action Deposit Modificado: L:AUTOR,R:PROFESSORA

Portanto, tal como indicado pelo *TODO* provido pela professora, o autor produziu os seguintes casos, dos quais se destituía o trabalho primordialmente, se baseando no formalismo propiciado pela mesma em linguagem Quint:



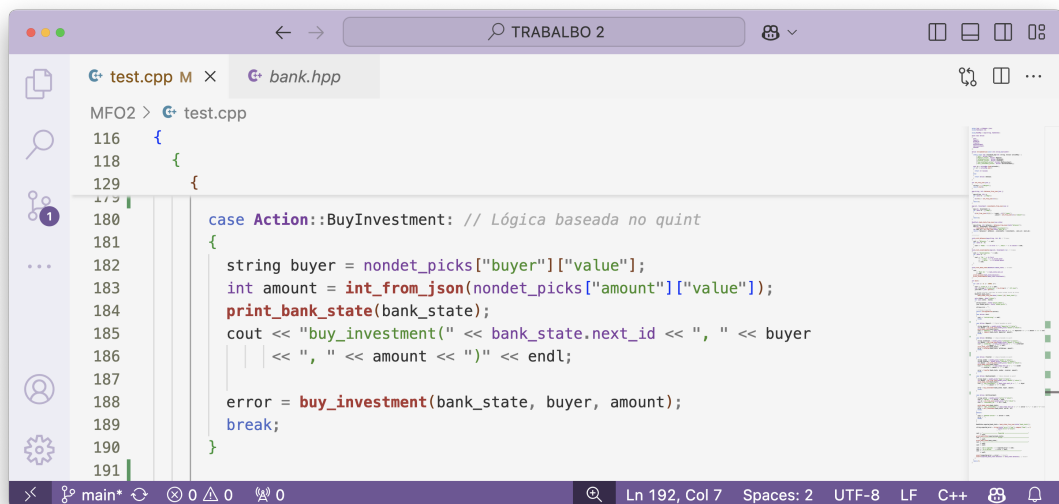


The screenshot shows a C++ IDE with two tabs: 'test.cpp' and 'bank.hpp'. The 'test.cpp' file is open, showing a switch statement. The current case is 'case Action::Transfer: // Lógica baseada no quint'. The code inside the case block is as follows:

```
116 {
118 {
129 {
167 case Action::Transfer: // Lógica baseada no quint
168 {
169     string sender = nondet_picks["sender"]["value"];
170     string receiver = nondet_picks["receiver"]["value"];
171     int amount = int_from_json(nondet_picks["amount"]["value"]);
172     print_bank_state(bank_state);
173     cout << "transfer(" << bank_state.next_id << ", " << sender
174         << receiver << amount << ")" << endl;
175
176     error = transfer(bank_state, sender, receiver, amount);
177     break;
178 }
```

The status bar at the bottom indicates 'Ln 179, Col 7', 'Spaces: 2', 'UTF-8', 'LF', and 'C++'.

Figura 4. Case Action Transfer



The screenshot shows the same C++ IDE with the 'test.cpp' file open. The current case is 'case Action::BuyInvestment: // Lógica baseada no quint'. The code inside the case block is as follows:

```
180 case Action::BuyInvestment: // Lógica baseada no quint
181 {
182     string buyer = nondet_picks["buyer"]["value"];
183     int amount = int_from_json(nondet_picks["amount"]["value"]);
184     print_bank_state(bank_state);
185     cout << "buy_investment(" << bank_state.next_id << ", " << buyer
186         << ", " << amount << ")" << endl;
187
188     error = buy_investment(bank_state, buyer, amount);
189     break;
190 }
191 }
```

The status bar at the bottom indicates 'Ln 192, Col 7', 'Spaces: 2', 'UTF-8', 'LF', and 'C++'.

Figura 5. Case Action Buylnestment

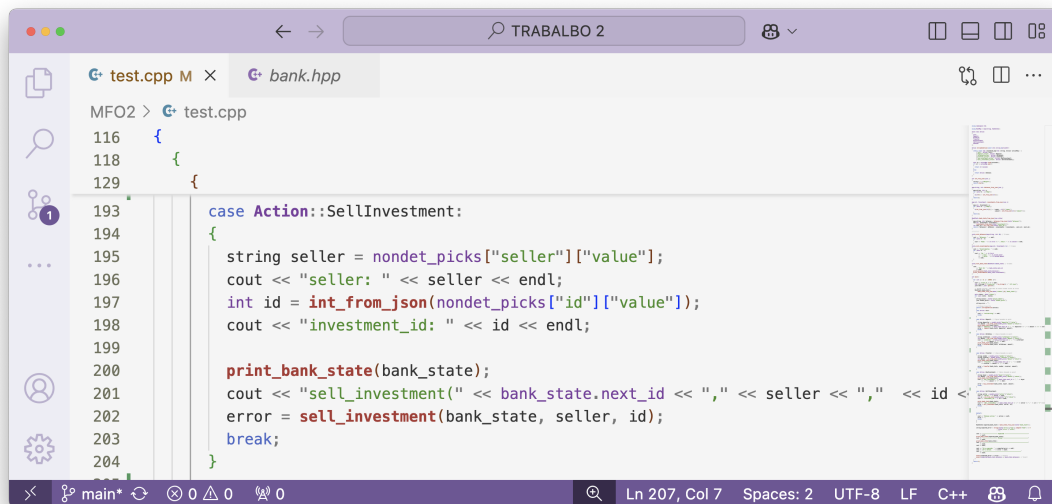


Figura 6. Case Action SellInvestment

Ao fim da corrente seção, apresenta-se o mais extremadamente modificado fragmento do código: o caso *default*.

Nesse, para fins de *debug*, foram produzidos *feedbacks* usando a função `std::cout` e as funções de autoria própria do autor apresentadas na Figura 1, que auxiliou na verificação de resultados preliminares, até que fosse, enfim, atingido o esperado.

Finalmente, no intuito de satisfazer a questão "TODO: comparar o erro esperado com o erro obtido" foram implementados *asserts*, que têm como intento assemelharem-se às invariantes de métodos formais.

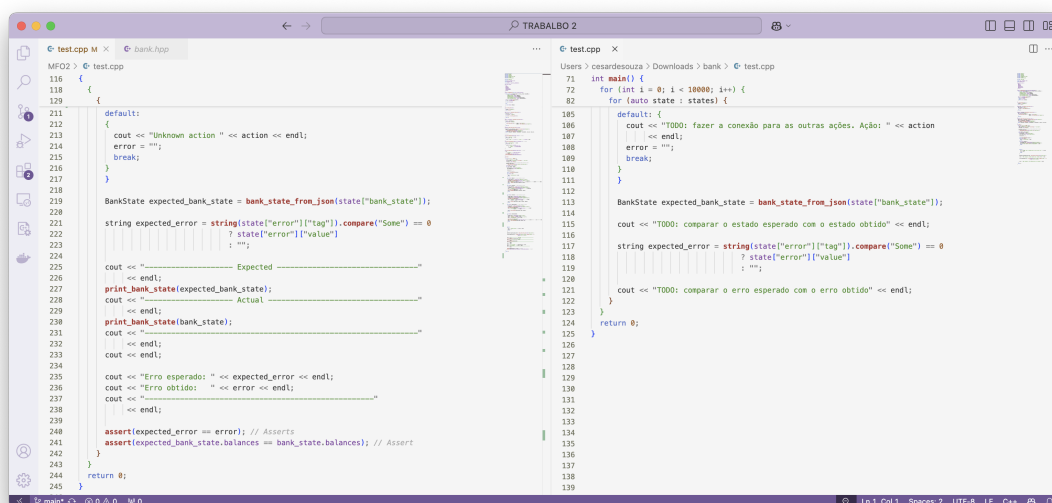
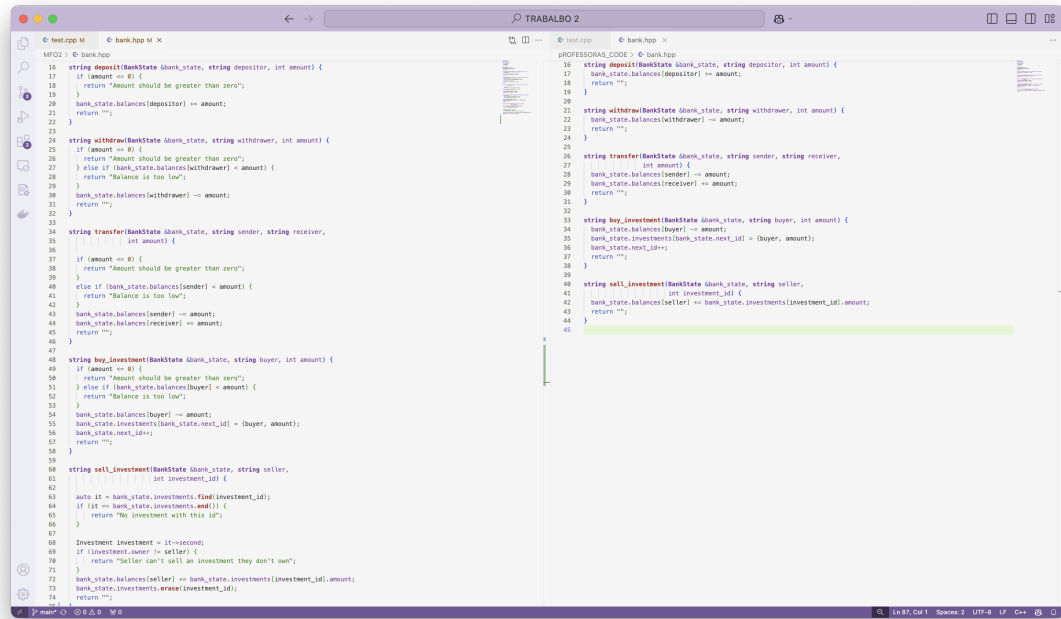


Figura 7. Case default: L:AUTOR,R:PROFESSORA

2.1. Cabeçalho (bank.hpp)

É possível atestar modificações por todo o arquivo, por exemplo na função *sell_investment()*, permitindo que seja performedo uma deleção no investimento, afinal o tal teria sido vendido. Segue em captura de tela.



```
16 string deposit(BankState &bank_state, string depositor, int amount) {
17     if (amount <= 0) {
18         return "Amount should be greater than zero";
19     }
20     bank_state.balances[depositor] += amount;
21     return "";
22 }
23
24 string withdraw(BankState &bank_state, string withdrawer, int amount) {
25     if (amount <= 0) {
26         return "Amount should be greater than zero";
27     } else if (bank_state.balances[withdrawer] < amount) {
28         return "Balance is too low";
29     }
30     bank_state.balances[withdrawer] -= amount;
31     return "";
32 }
33
34 string transfer(BankState &bank_state, string sender, string receiver,
35               int amount) {
36     if (amount <= 0) {
37         return "Amount should be greater than zero";
38     }
39     else if (bank_state.balances[sender] < amount) {
40         return "Balance is too low";
41     }
42     bank_state.balances[sender] -= amount;
43     bank_state.balances[receiver] += amount;
44     return "";
45 }
46
47 string buy_investment(BankState &bank_state, string buyer, int amount) {
48     if (amount <= 0) {
49         return "Amount should be greater than zero";
50     } else if (bank_state.balances[buyer] < amount) {
51         return "Balance is too low";
52     }
53     bank_state.balances[buyer] -= amount;
54     bank_state.investments[bank_state.next_id] = {buyer, amount};
55     bank_state.next_id++;
56     return "";
57 }
58
59 string sell_investment(BankState &bank_state, string seller,
60                       int investment_id) {
61     auto it = bank_state.investments.find(investment_id);
62     if (it == bank_state.investments.end()) {
63         return "No investment with this id";
64     }
65     Investment investment = it->second;
66     if (investment.owner != seller) {
67         return "Seller can't sell an investment they don't own";
68     }
69     bank_state.balances[seller] += bank_state.investments[investment_id].amount;
70     bank_state.investments.erase(investment_id);
71     return "";
72 }
```

Figura 8. bank.hpp: L:AUTOR,R:PROFESSORA

