

# **LingPON 2.0 & compilador para solução PON em C++ orientado a espaço de nomes**

Larissa Keiko Oshiro, (L. F. Pordeus, A. F. Ronszcka), [J. A. Fabro, J. M. Simão]

# LingPon 2.0 e Implementação Namespaces C++

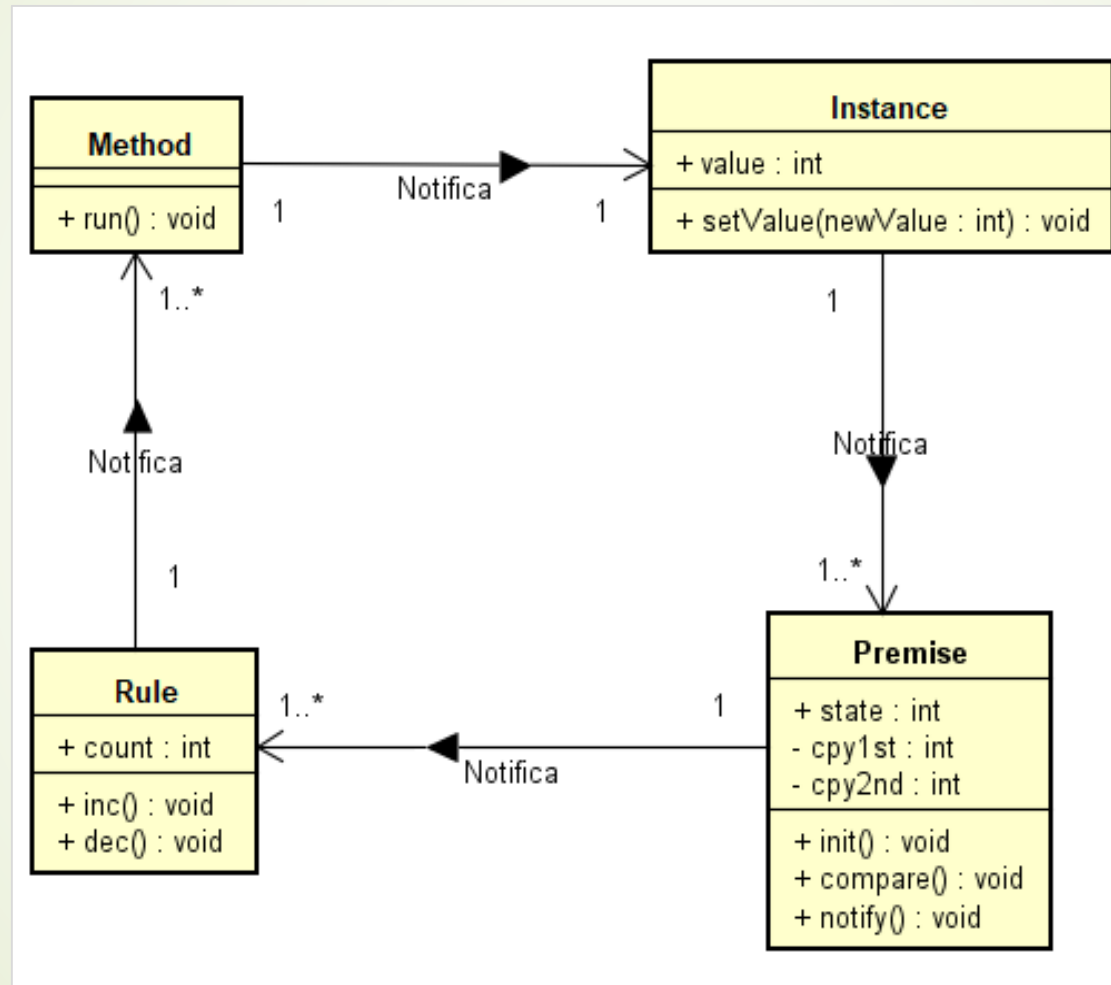


Figura 1: Materialização da geração de código do LingPon em namespaces C++

Fonte: Adaptado de Negrini (2016)

```

1 fbe Main
2
3 includes FRAMEWORK_CPP_2_0
4     #include "SMSSender.h"
5     #include <iostream>
6     using namespace std;
7 end_includes
8
9 private Sector sectorA
10 private Sector sectorB
11
12 private method mtSendSms
13     params
14         String cellphone
15     end_params
16     code FRAMEWORK_CPP_2_0
17         SMSSender *sender = new SMSSender();
18         sender->send(cellphone);
19     end_code
20 end_method
21
22 rule rInvasionDetection
23     condition
24         premise prSectorAInvaded
25             sectorA.atIntruderDetected == true
26         end_premise
27         or
28         premise prSectorBInvaded
29             sectorB.atIntruderDetected == true
30         end_premise
31     end_condition
32     action sequential
33         instigation
34             call this.mtSendSms("41-999999999", "47-999999999")
35         end_instigation
36     end_action
37 end_rule
38
39 properties
40     strategy PRIORITY
41 end_properties
42
43 end_fbe

```

Figura 2: Projeto Sensors – Aplicação em Lingpon 2.0 utilizada para a geração de código em namespaces

# Namespace Instances

```
1 #include "instances.h"
2 #include "premises.h"
3 #include <string>
4
5 #include "SMSSender.h"
6 #include <iostream>
7 using namespace std;
8
9 namespace instance{
10 namespace sectorA{
11 namespace at{
12 namespace atIntruderDetected{
13 bool value = 0;
14 void setValue(bool newValue){
15     if (value != newValue){
16         value = newValue;
17         premise::main::prSectorAInvaded::notify_sectorA_atIntruderDetected(newValue);
18         premise::sectorA::prSectorInPeaceA::notify_sectorB_atIntruderDetected(newValue);
19         premise::sectorA::prSectorInPeaceB::notify_sectorB_atIntruderDetected(newValue);
20     }
21 }
22 }
23 }
24 namespace alarmA{
25 namespace at{
26 namespace atStatus{
27 bool value = 0;
28 void setValue(bool newValue){
29     if (value != newValue){
30         value = newValue;
31         premise::sectorA::prAlarmAOn::notify_alarmA_atStatus(newValue);
32     }
33 }
34 }
35 }
36 }
37 namespace alarmB{
38 namespace at{
39 namespace atStatus{
40 bool value = 0;
41 void setValue(bool newValue){
42     if (value != newValue){
43         value = newValue;
44         premise::sectorA::prAlarmBOn::notify_alarmB_atStatus(newValue);
45     }
46 }
47 }
48 }
49 }
50 namespace sensorA1{
51 namespace at{
52 namespace atState{
53 bool value = 0;
54 void setValue(bool newValue){
55     if (value != newValue){
56         value = newValue;
57         premise::sectorA::prSensorA1State::notify_sensorA1_atState(newValue);
58     }
59 }
60 }
61 }
62 }
63 namespace sensorA2{
64 namespace at{
65 namespace atState{
66 bool value = 0;
67 void setValue(bool newValue){
68     if (value != newValue){
69         value = newValue;
70         premise::sectorA::prSensorA2State::notify_sensorA2_atState(newValue);
71     }
72 }
73 }
74 }
75 }
76 namespace sensorB1{
77 namespace at{
78 namespace atState{
79 bool value = 0;
80 void setValue(bool newValue){
81     if (value != newValue){
82         value = newValue;
83         premise::sectorA::prSensorB1State::notify_sensorB1_atState(newValue);
84     }
85 }
86 }
87 }
88 }
89 namespace sirenA1{
90 namespace at{
91 namespace atTime{
92 int value = 0;
93 void setValue(int newValue){
94     if (value != newValue){
95         value = newValue;
96     }
97 }
98 }
99 }
100 }
101 namespace sirenA2{
102 namespace at{
103 namespace atTime{
104 int value = 0;
105 void setValue(int newValue){
106     if (value != newValue){
107         value = newValue;
108     }
109 }
110 }
111 }
112 }
113 namespace sirenB1{
114 namespace at{
115 namespace atTime{
116 int value = 0;
117 void setValue(int newValue){
118     if (value != newValue){
```

Figura 3: Implementação das Instances em namespaces

# Namespace Instances

```
119         value = newValue;
120     }
121 }
122 }
123 }
124 }
125 }
126 namespace sectorB{
127     namespace at{
128         namespace atIntruderDetected{
129             bool value = 0;
130             void setValue(bool newValue){
131                 if (value != newValue){
132                     value = newValue;
133                     premise::main::prSectorBInvaded::notify_sectorB_atIntruderDetected(newValue);
134                     premise::sectorB::prSectorInPeaceA::notify_sectorB_atIntruderDetected(newValue);
135                     premise::sectorB::prSectorInPeaceB::notify_sectorB_atIntruderDetected(newValue);
136                 }
137             }
138         }
139     }
140 }
141 namespace alarmA{
142     namespace at{
143         namespace atStatus{
144             bool value = 0;
145             void setValue(bool newValue){
146                 if (value != newValue){
147                     value = newValue;
148                     premise::sectorB::prAlarmA0n::notify_alarmA_atStatus(newValue);
149                 }
150             }
151         }
152     }
153 }
154 namespace alarmB{
155     namespace at{
156         namespace atStatus{
157             bool value = 0;
158             void setValue(bool newValue){
159                 if (value != newValue){
160                     value = newValue;
161                     premise::sectorB::prAlarmB0n::notify_alarmB_atStatus(newValue);
162                 }
163             }
164         }
165     }
166 }
167 namespace sensorA1{
168     namespace at{
169         namespace atState{
170             bool value = 0;
171             void setValue(bool newValue){
172                 if (value != newValue){
173                     value = newValue;
174                     premise::sectorB::prSensorA1State::notify_sensorA1_atState(newValue);
175                 }
176             }
177         }
178     }
179 }
```

```
178 }
179 namespace sensorA2{
180     namespace at{
181         namespace atState{
182             bool value = 0;
183             void setValue(bool newValue){
184                 if (value != newValue){
185                     value = newValue;
186                     premise::sectorB::prSensorA2State::notify_sensorA2_atState(newValue);
187                 }
188             }
189         }
190     }
191 }
192 }
193 namespace sensorB1{
194     namespace at{
195         namespace atState{
196             bool value = 0;
197             void setValue(bool newValue){
198                 if (value != newValue){
199                     value = newValue;
200                     premise::sectorB::prSensorB1State::notify_sensorB1_atState(newValue);
201                 }
202             }
203         }
204     }
205 }
206 }
207 namespace sirenA1{
208     namespace at{
209         namespace atTime{
210             int value = 0;
211             void setValue(int newValue){
212                 if (value != newValue){
213                     value = newValue;
214                 }
215             }
216         }
217     }
218 }
219 }
220 namespace sirenA2{
221     namespace at{
222         namespace atTime{
223             int value = 0;
224             void setValue(int newValue){
225                 if (value != newValue){
226                     value = newValue;
227                 }
228             }
229         }
230     }
231 }
232 }
233 namespace sirenB1{
234     namespace at{
235         namespace atTime{
236             int value = 0;
237             void setValue(int newValue){
238                 if (value != newValue){
239                     value = newValue;
240                 }
241             }
242         }
243     }
244 }
```

Figura 4: Implementação das Instâncias em namespaces

# Namespace Premises

```
1 #include "premises.h"
2 #include "rules.h"
3 #include <string>
4
5 #include "SMSSender.h"
6 #include <iostream>
7 using namespace std;
8
9 namespace premise{
10     namespace sectorA{
11         namespace prAlarmA0n{
12             bool state = false;
13             bool cpy1st, cpy2nd;
14             void init(){
15                 cpy1st = 0;
16                 cpy2nd = 1;
17             }
18             void compare(){
19                 if(cpy1st == cpy2nd){
20                     if(state == false){
21                         state = true;
22                         rule::sectorA::r1FireAlarmA::incl();
23                     }
24                 }else{
25                     if(state == true){
26                         state = false;
27                         rule::sectorA::r1FireAlarmA::decl();
28                     }
29                 }
30             }
31             void notify_alarmA_atStatus(bool newValue){
32                 cpy1st = newValue;
33                 compare();
34             }
35             // Notified by attributes: [ atStatus ]
36         }
37         namespace prSectorInPeaceA{
38             bool state = false;
39             bool cpy1st, cpy2nd;
40             void init(){
41                 cpy1st = 0;
42                 cpy2nd = 0;
43             }
44             void compare(){
45                 if(cpy1st == cpy2nd){
46                     if(state == false){
47                         state = true;
48                         rule::sectorA::r1FireAlarmA::incl();
49                     }
50                 }else{
51                     if(state == true){
52                         state = false;
53                         rule::sectorA::r1FireAlarmA::decl();
54                     }
55                 }
56             }
57             void notify_sectorB_atIntruderDetected(bool newValue){
58                 cpy1st = newValue;
59                 compare();
60             }
61             // Notified by attributes: [ atIntruderDetected ]
62         }
63     }
64     namespace prSensorA1State{
65         bool state = false;
66         bool cpy1st, cpy2nd;
67         void init(){
68             cpy1st = 0;
69             cpy2nd = 1;
70         }
71         void compare(){
72             if(cpy1st == cpy2nd){
73                 if(state == false){
74                     state = true;
75                     rule::sectorA::r1FireAlarmA::inc2();
76                 }
77             }else{
78                 if(state == true){
79                     state = false;
80                     rule::sectorA::r1FireAlarmA::dec2();
81                 }
82             }
83         }
84         void notify_sensorA1_atState(bool newValue){
85             cpy1st = newValue;
86             compare();
87         }
88         // Notified by attributes: [ atState ]
89     }
90     namespace prSensorA2State{
91         bool state = false;
92         bool cpy1st, cpy2nd;
93         void init(){
94             cpy1st = 0;
95             cpy2nd = 1;
96         }
97         void compare(){
98             if(cpy1st == cpy2nd){
99                 if(state == false){
100                     state = true;
101                     rule::sectorA::r1FireAlarmA::inc2();
102                 }
103             }else{
104                 if(state == true){
105                     state = false;
106                     rule::sectorA::r1FireAlarmA::dec2();
107                 }
108             }
109         }
110         void notify_sensorA2_atState(bool newValue){
111             cpy1st = newValue;
112             compare();
113         }
114         // Notified by attributes: [ atState ]
115     }
116     namespace prAlarmB0n{
117         bool state = false;
118         bool cpy1st, cpy2nd;
119         void init(){
```

Figura 5: Implementação das Premises em namespaces



# Namespace Premises

```
119     cpy1st = 0;
120     cpy2nd = 1;
121 }
122 void compare(){
123     if(cpy1st == cpy2nd){
124         if(state == false){
125             state = true;
126             rule::sectorA::rlFireAlarmB::inc();
127         }
128     }else{
129         if(state == true){
130             state = false;
131             rule::sectorA::rlFireAlarmB::dec();
132         }
133     }
134 }
135 void notify_alarmB_atStatus(bool newValue){
136     cpy1st = newValue;
137     compare();
138 }
139 // Notified by attributes: [ atStatus ]
140 }
141 namespace prSectorInPeaceB{
142     bool state = false;
143     bool cpy1st, cpy2nd;
144     void init(){
145         cpy1st = 0;
146         cpy2nd = 0;
147     }
148     void compare(){
149         if(cpy1st == cpy2nd){
150             if(state == false){
151                 state = true;
152                 rule::sectorA::rlFireAlarmB::inc();
153             }
154         }else{
155             if(state == true){
156                 state = false;
157                 rule::sectorA::rlFireAlarmB::dec();
158             }
159         }
160     }
161     void notify_sectorB_atIntruderDetected(bool newValue){
162         cpy1st = newValue;
163         compare();
164     }
165     // Notified by attributes: [ atIntruderDetected ]
166 }
167 namespace prSensorB1State{
168     bool state = false;
169     bool cpy1st, cpy2nd;
170     void init(){
171         cpy1st = 0;
172         cpy2nd = 1;
173     }
174     void compare(){
175         if(cpy1st == cpy2nd){
176             if(state == false){
177                 state = true;
377     }
378     namespace main{
379         namespace prSectorAInvaded{
380             bool state = false;
381             bool cpy1st, cpy2nd;
382             void init(){
383                 cpy1st = 0;
384                 cpy2nd = 1;
385             }
386             void compare(){
387                 if(cpy1st == cpy2nd){
388                     if(state == false){
389                         state = true;
390                         rule::main::rlInvasionDetection::inc();
391                     }
392                 }else{
393                     if(state == true){
394                         state = false;
395                         rule::main::rlInvasionDetection::dec();
396                     }
397                 }
398             }
399             void notify_sectorA_atIntruderDetected(bool newValue){
400                 cpy1st = newValue;
401                 compare();
402             }
403             // Notified by attributes: [ atIntruderDetected ]
404         }
405         namespace prSectorBInvaded{
406             bool state = false;
407             bool cpy1st, cpy2nd;
408             void init(){
409                 cpy1st = 0;
410                 cpy2nd = 1;
411             }
412             void compare(){
413                 if(cpy1st == cpy2nd){
414                     if(state == false){
415                         state = true;
416                         rule::main::rlInvasionDetection::inc();
417                     }
418                 }else{
419                     if(state == true){
420                         state = false;
421                         rule::main::rlInvasionDetection::dec();
422                     }
423                 }
424             }
425             void notify_sectorB_atIntruderDetected(bool newValue){
426                 cpy1st = newValue;
427                 compare();
428             }
429             // Notified by attributes: [ atIntruderDetected ]
430         }
431     }
432 }
433 }
```

Figura 6: Implementação das Premises em namespaces

# Namespace Rules

```
1 #include "rules.h"
2 #include "methods.h"
3 #include <string>
4
5     #include "SMSSender.h"
6     #include <iostream>
7     using namespace std;
8
9 namespace rule{
10     namespace main{
11         namespace r1InvasionDetection{
12             int count = 0;
13             void inc(){
14                 count++;
15                 if (count >= 1){
16                     method::main::mtSendSms::mtSendSms("41-999999999");
17                     method::main::mtSendSms::mtSendSms("47-999999999");
18                 }
19             }
20             void dec(){
21                 count--;
22             }
23         }
24     }
25     namespace sectorA{
26         namespace r1FireAlarmA{
27             int count1 = 0;
28             bool status1;
29             int count2 = 0;
30             bool status2;
31             void incl(){
32                 count1++;
33                 status1 = false;
34                 if (count1 = 2){
35                     status1 = true;
36                     compareStatusSubConditions();
37                 }
38             }
39             void decl(){
40                 count1--;
41             }
42             void inc2(){
43                 count2++;
44                 status2 = false;
45                 if (count2 >= 1){
46                     status2 = true;
47                     compareStatusSubConditions();
48                 }
49             }
50             void dec2(){
51                 count2--;
52             }
53             void compareStatusSubConditions(){
54                 if((status1 = true) && (status2 = true)){
55                     //method::sirenA1::mtFire::mtFire(10);
56                     //method::sirenA2::mtFire::mtFire(30);
57                     method::sectorA::mtNotifyInvasion::mtNotifyInvasion();
58                 }
59             }
60         }
61         namespace r1FireAlarmB{
62             int count = 0;
63             void inc(){
64                 count++;
65                 if (count = 3){
66                     //method::sirenB1::mtFire::mtFire(10);
67                     method::sectorA::mtNotifyInvasion::mtNotifyInvasion();
68                 }
69             }
70             void dec(){
71                 count--;
72             }
73         }
74     }
75     namespace sectorB{
76         namespace r1FireAlarmA{
77             int count1 = 0;
78             bool status1;
79             int count2 = 0;
80             bool status2;
81             void incl(){
82                 count1++;
83                 status1 = false;
84                 if (count1 = 2){
85                     status1 = true;
86                     compareStatusSubConditions();
87                 }
88             }
89             void decl(){
90                 count1--;
91             }
92             void inc2(){
93                 count2++;
94                 status2 = false;
95                 if (count2 >= 1){
96                     status2 = true;
97                     compareStatusSubConditions();
98                 }
99             }
100             void dec2(){
101                 count2--;
102             }
103             void compareStatusSubConditions(){
104                 if((status1 = true) && (status2 = true)){
105                     //method::sirenA1::mtFire::mtFire(10);
106                     //method::sirenA2::mtFire::mtFire(30);
107                     method::sectorB::mtNotifyInvasion::mtNotifyInvasion();
108                 }
109             }
110         }
111         namespace r1FireAlarmB{
112             int count = 0;
113             void inc(){
114                 count++;
115                 if (count = 3){
116                     //method::sirenB1::mtFire::mtFire(10);
117                     method::sectorB::mtNotifyInvasion::mtNotifyInvasion();
118                 }
119             }
120         }
121     }
122 }
```

Figura 7: Implementação das Rules em namespaces



# Namespace Methods

```
1 #include "methods.h"
2 #include "instances.h"
3 #include <string>
4
5     #include "SMSSender.h"
6     #include <iostream>
7     using namespace std;
8
9 namespace method{
10     namespace main{
11         namespace mtSendSms{
12             void mtSendSms(std::string cellphone){
13
14                 SMSSender *sender = new SMSSender();
15                 sender->send(cellphone);
16
17             }
18         }
19     }
20     namespace sectorA{
21         namespace mtNotifyInvasion{
22             void mtNotifyInvasion(){
23                 instance::sectorA::at::atIntruderDetected::setValue(1);
24             }
25         }
26     }
27     namespace sectorB{
28         namespace mtNotifyInvasion{
29             void mtNotifyInvasion(){
30                 instance::sectorB::at::atIntruderDetected::setValue(1);
31             }
32         }
33     }
34 }
35
```

Figura 8: Implementação dos Methods em namespaces

# Class SMSSender

SMSSender.h	SMSSender.cpp
<pre>1 #include &lt;iostream&gt; 2 #include &lt;string&gt; 3 using namespace std; 4 5 class SMSSender 6 { 7     std::string telephone; 8 9 public: 10     //inicializa(); 11     void send(std::string tel); 12 };</pre>	<pre>1 #include "SMSSender.h" 2 #include &lt;string&gt; 3 using namespace std; 4 5 void SMSSender::send(std::string tel) 6 { 7     telephone = tel; 8     cout &lt;&lt; "Enviando mensagem para " &lt;&lt; telephone &lt;&lt; "... " &lt;&lt; endl; 9 };</pre>

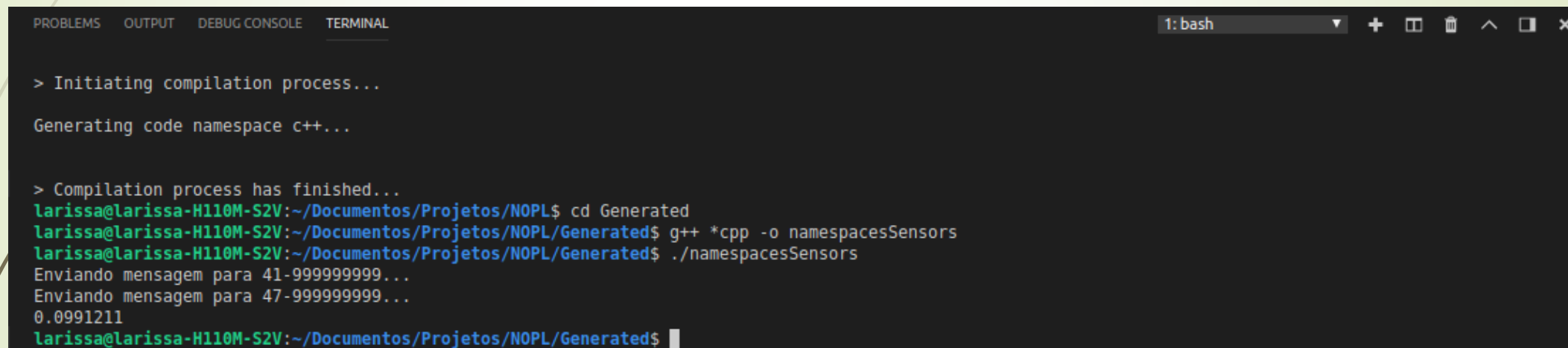
Figura 9: Exemplo de Classe externa para a inclusão de bibliotecas em IncludeBlock

# Main

```
1 #include <stdio.h>
2 #include <sys/time.h>
3 #include <iostream>
4 #include "premises.h"
5 #include "instances.h"
6
7 using namespace std;
8
9 int main() {
10     premise::sectorA::prAlarmAOn::init();
11     premise::sectorA::prSectorInPeaceA::init();
12     premise::sectorA::prSensorA1State::init();
13     premise::sectorA::prSensorA2State::init();
14     premise::sectorA::prAlarmBOn::init();
15     premise::sectorA::prSectorInPeaceB::init();
16     premise::sectorA::prSensorB1State::init();
17     premise::sectorB::prAlarmAOn::init();
18     premise::sectorB::prSectorInPeaceA::init();
19     premise::sectorB::prSensorA1State::init();
20     premise::sectorB::prSensorA2State::init();
21     premise::sectorB::prAlarmBOn::init();
22     premise::sectorB::prSectorInPeaceB::init();
23     premise::sectorB::prSensorB1State::init();
24     premise::main::prSectorAInvaded::init();
25     premise::main::prSectorBInvaded::init();
26
27     timeval time;
28     double initial;
29     double final;
30     gettimeofday(&time,0);
31     initial = (time.tv_sec * 1000.0) + (time.tv_usec / 1000.0);
32
33     instance::sectorB::at::atIntruderDetected::setValue(0);
34     instance::sectorB::alarmB::at::atStatus::setValue(1);
35     instance::sectorB::sensorB1::at::atState::setValue(1);
36
37     gettimeofday(&time,0);
38     final = (time.tv_sec * 1000.0) + (time.tv_usec / 1000.0);
39     double resultado = final - initial;
40     cout << resultado << endl;
41     return 0;
42 }
43
```

Figura 10: Implementação da Main.cpp em namespaces

# Testes e Resultados



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash + [ ] [ ] [ ] [ ] X  
  
> Initiating compilation process...  
Generating code namespace c++...  
  
> Compilation process has finished...  
larissa@larissa-H110M-S2V:~/Documentos/Projetos/NOPL$ cd Generated  
larissa@larissa-H110M-S2V:~/Documentos/Projetos/NOPL/Generated$ g++ *cpp -o namespacesSensors  
larissa@larissa-H110M-S2V:~/Documentos/Projetos/NOPL/Generated$ ./namespacesSensors  
Enviando mensagem para 41-999999999...  
Enviando mensagem para 47-999999999...  
0.0991211  
larissa@larissa-H110M-S2V:~/Documentos/Projetos/NOPL/Generated$
```

Figura 11: Resultados da geração de código de LingPon 2.0 em namespaces C++ (Projeto Sensors)

# Obrigada!

