Patterns of Algorithms - Examples

Example 1 – Sequence calculation

Average of marks: We know the student's marks from a given subject. Let's calculate the average of marks.

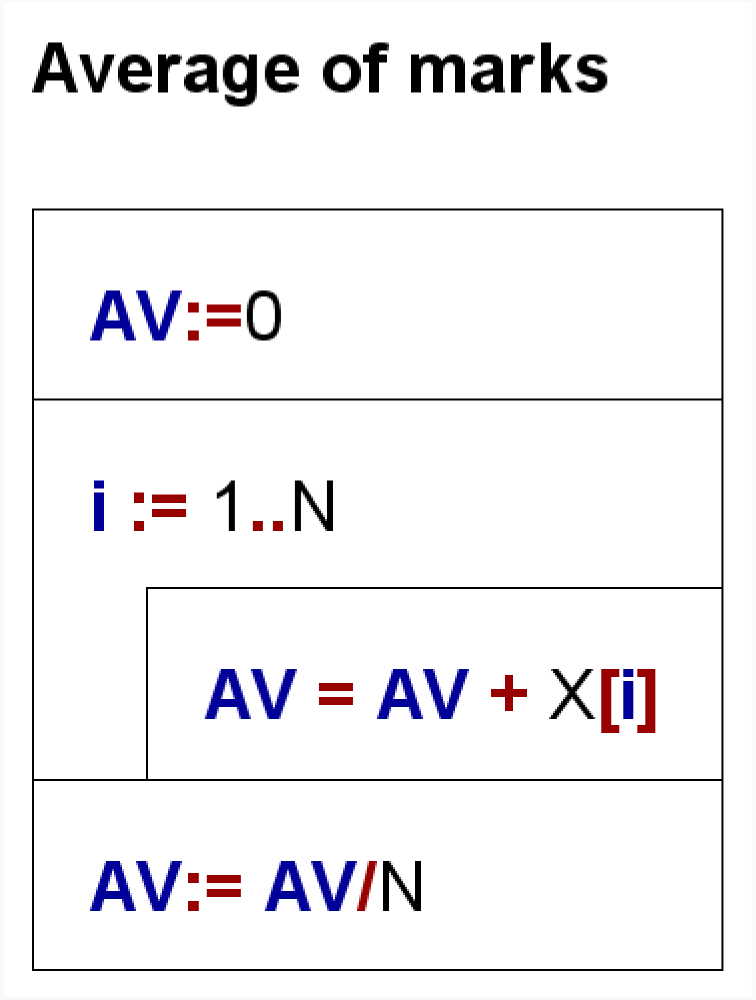
Specification:

Input: $N \in \mathbb{N}$, $X_{1..N} \in \mathbb{N}^N$

Output: $AV \in \mathbb{R}$

Precondition: $\forall\, i\ (1 \le i \le N) : X_i \in [1..5]$

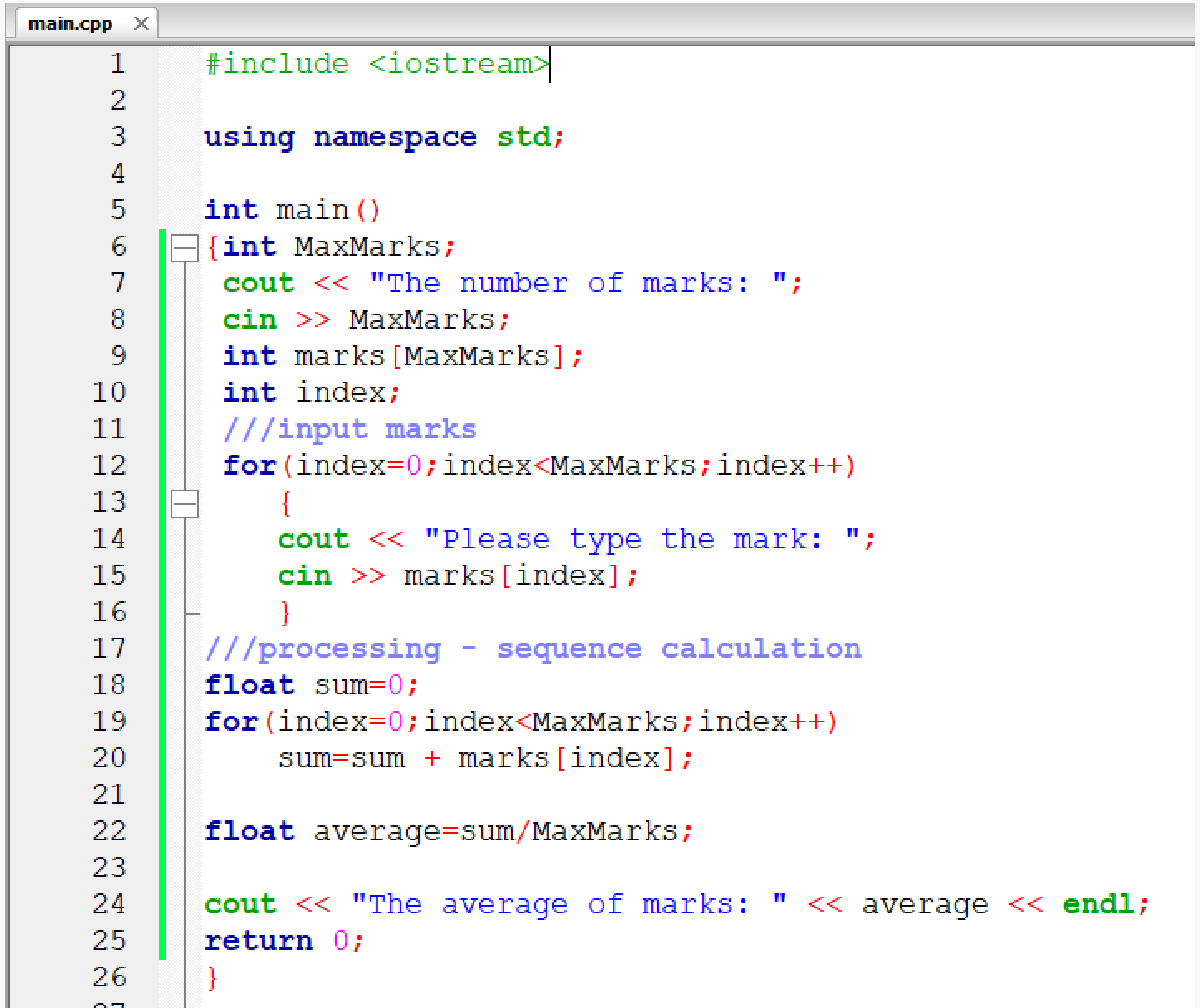
Postcondition: $AV = \dfrac{\sum_{i=1}^{N} X_i}{N}$



Average of marks

```
AV:=0
i := 1..N
    AV = AV + X[i]
AV:= AV/N
```

Source code 1 – Sequence calculation – Average of marks


main.cpp

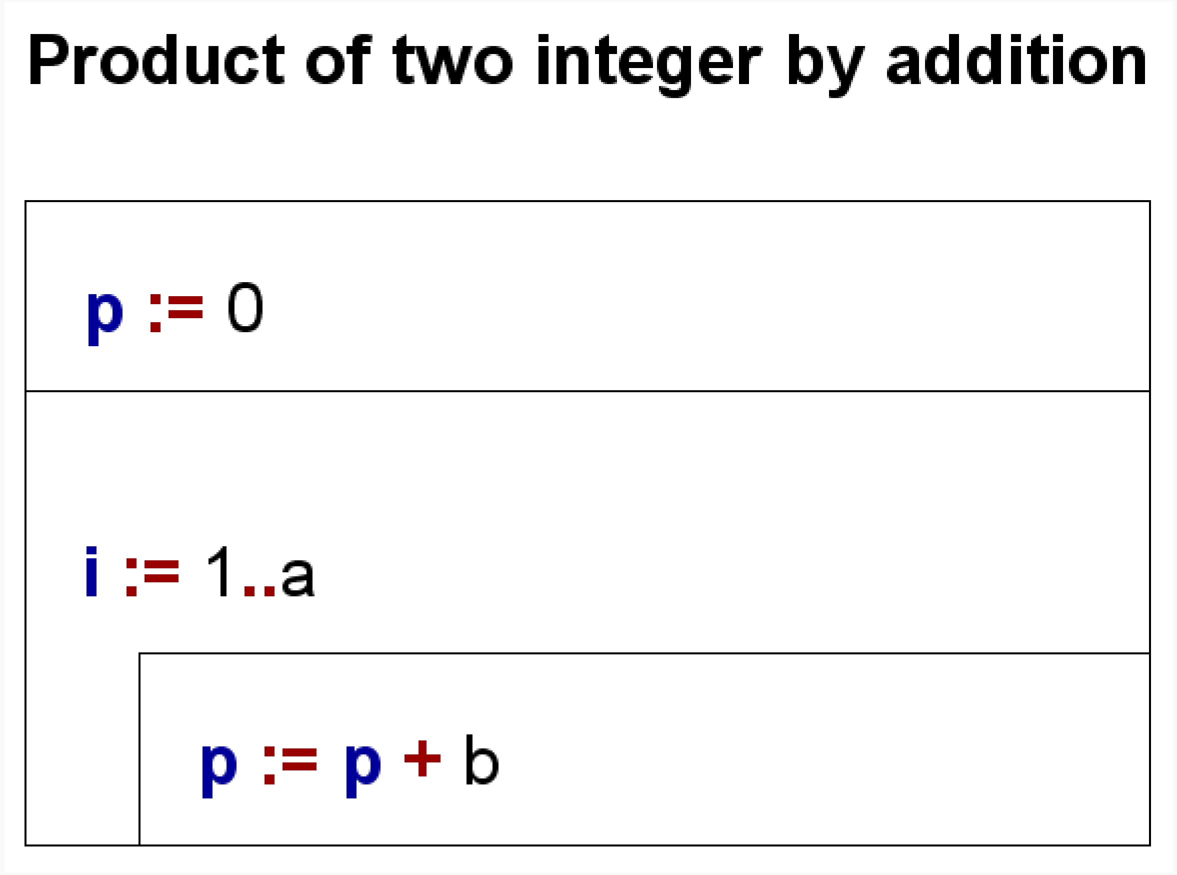
```cpp
#include <iostream>

using namespace std;

int main()
{int MaxMarks;
 cout << "The number of marks: ";
 cin >> MaxMarks;
 int marks[MaxMarks];
 int index;
 ///input marks
 for(index=0;index<MaxMarks;index++)
    {
    cout << "Please type the mark: ";
    cin >> marks[index];
    }
///processing - sequence calculation
float sum=0;
for(index=0;index<MaxMarks;index++)
    sum=sum + marks[index];

float average=sum/MaxMarks;

cout << "The average of marks: " << average << endl;
return 0;
}
```

Example 2 – Sequence calculation

Product of a and b by addition: Suppose our computer knows only one operation, that is addition. Let’s calculate the product of **a** and **b** by addition.

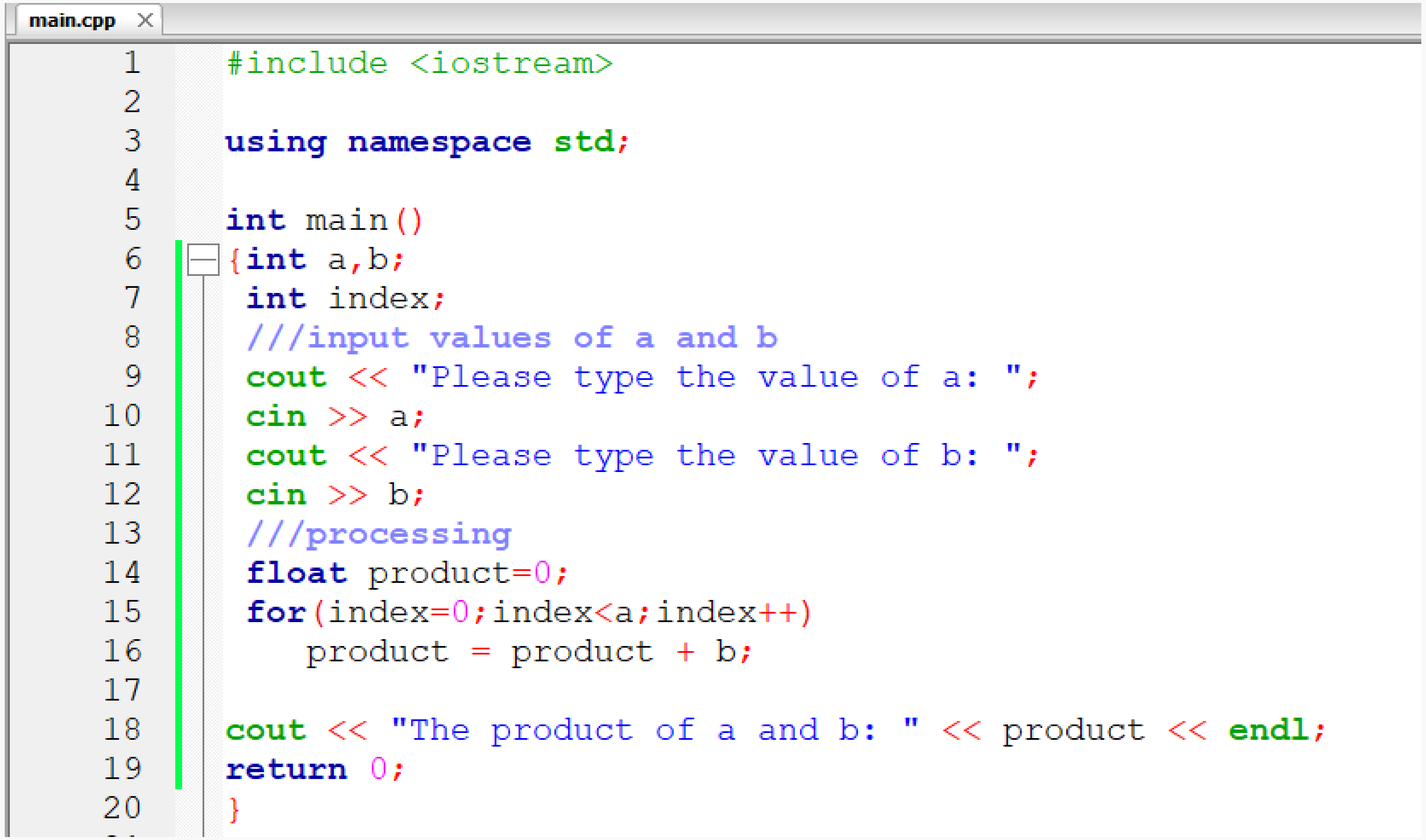
Specification:
Input: a, b $\in \mathbb{Z}$
Output: p $\in \mathbb{Z}$
Precondition: a, b $\neq 0$

Postcondition: p = $\sum_{i=1}^{a} b$


Product of two integer by addition

```
p := 0
i := 1..a
    p := p + b
```

Source code 2 – Sequence calculation – Product of a and b by addition



```cpp
#include <iostream>

using namespace std;

int main()
{int a,b;
 int index;
 ///input values of a and b
 cout << "Please type the value of a: ";
 cin >> a;
 cout << "Please type the value of b: ";
 cin >> b;
 ///processing
 float product=0;
 for(index=0;index<a;index++)
    product = product + b;

cout << "The product of a and b: " << product << endl;
return 0;
}
```

Example 3 – Counting

Number of proper numbers: Let’s count the numbers from 1 to 100 that’s are even and divisible by 7.

Specification:

Input: $N \in \mathbb{N}$, $X_{1..N} \in \mathbb{N}^N$

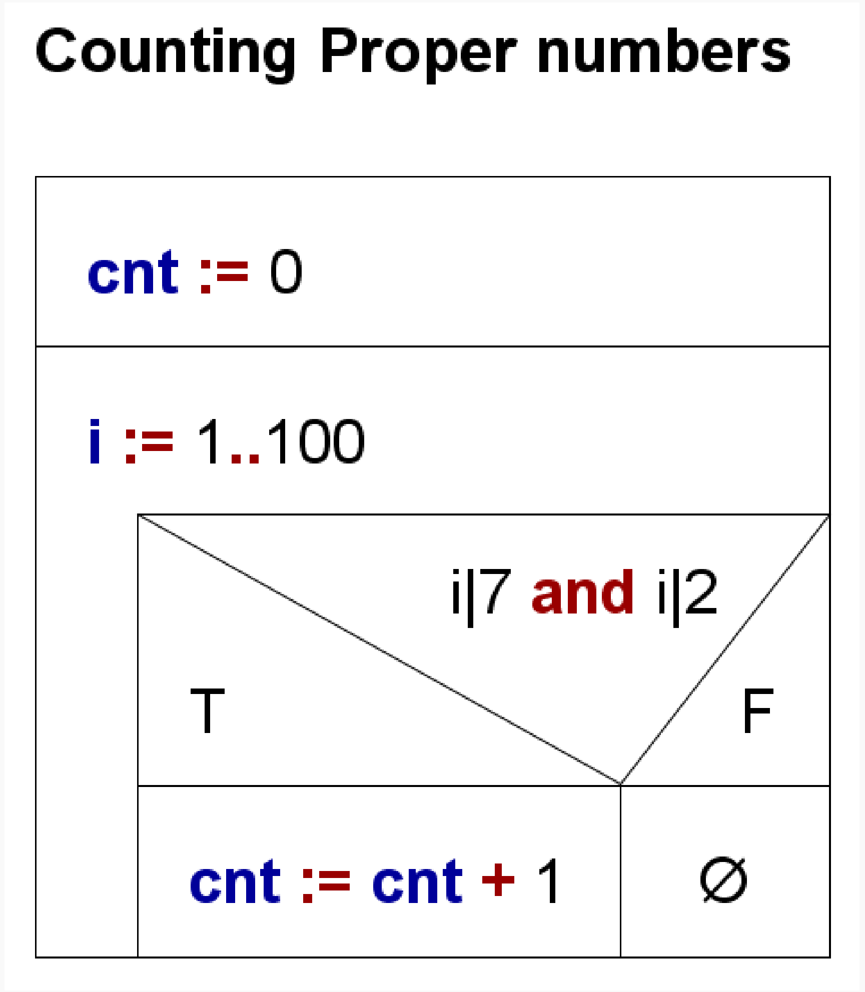
Proper number? $\mathbb{N} \to L$

Proper number? $(x) := (x \mid 7 \text{ and } x \mid 2)$

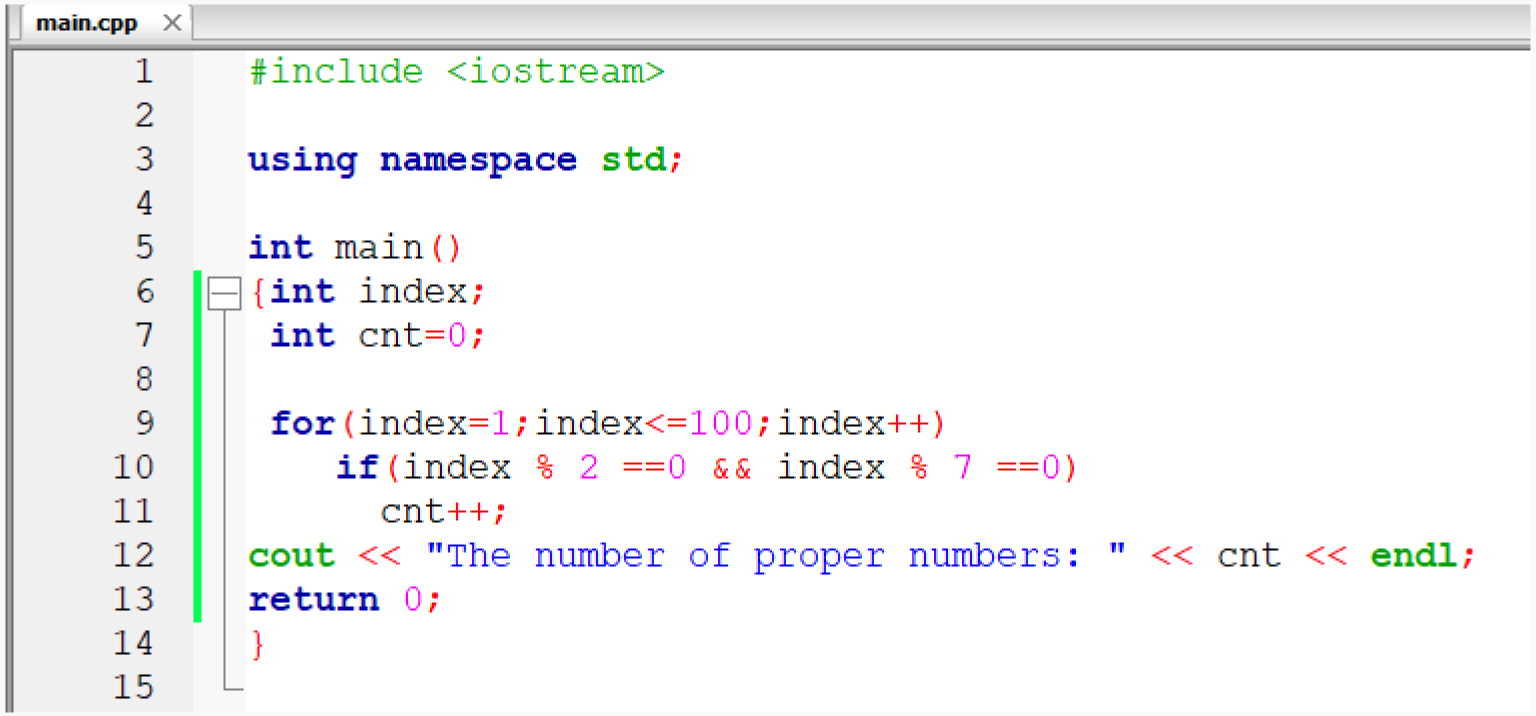
Output: $cnt \in \mathbb{N}$

Precondition: $\forall i \ (1 \leq i \leq 100) : X_i \in [1..100]$

Postcondition: $cnt = \sum_{\substack{i=1 \\ Proper\ number?(X_i)}}^{N} 1$



Source code 3 – Counting – Number of proper numbers



main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{int index;
 int cnt=0;

 for(index=1;index<=100;index++)
    if(index % 2 ==0 && index % 7 ==0)
      cnt++;
cout << "The number of proper numbers: " << cnt << endl;
return 0;
}

```

Example 4 – Maximum selection

The longest name: There is a list with the name of students. Let’s select the longest name from this list.

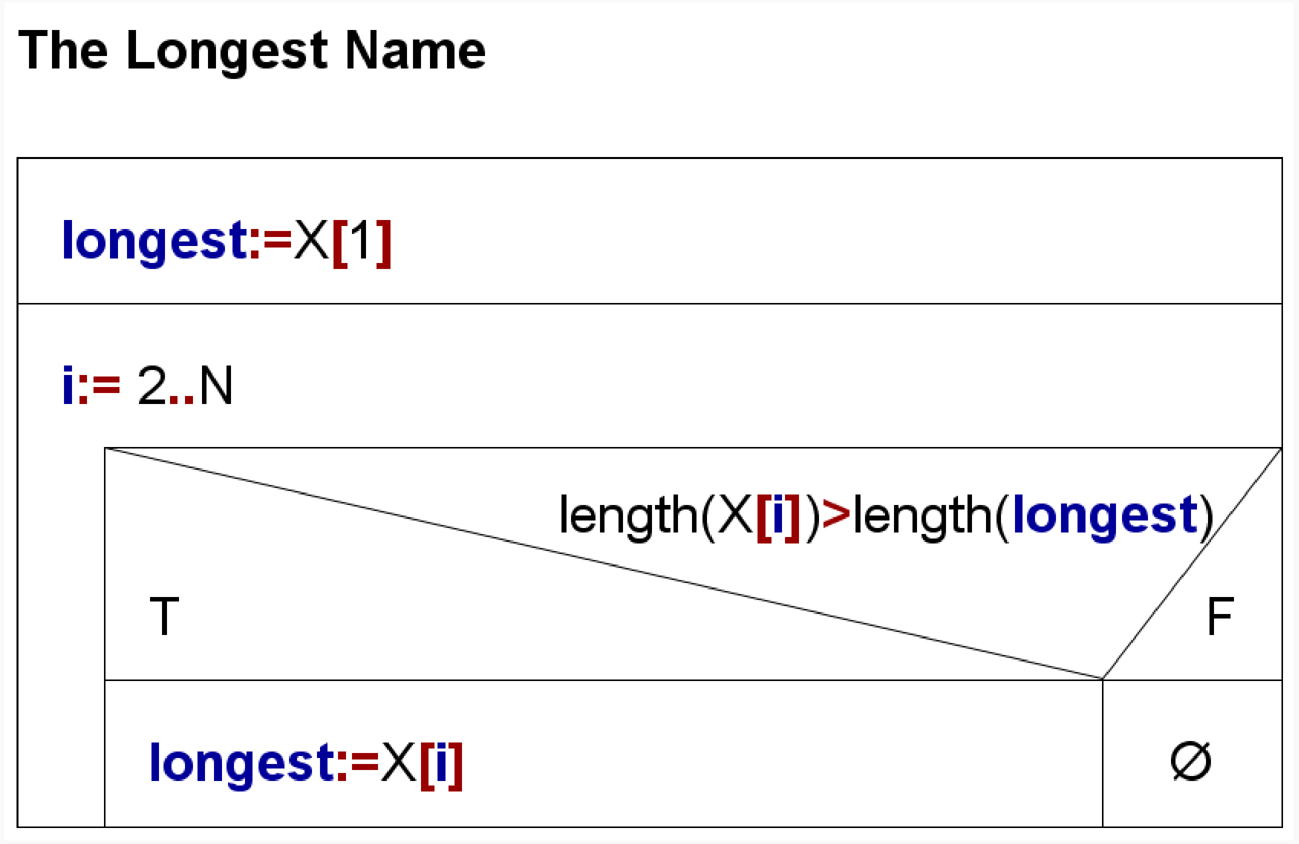
Specification:

Input: $N \in \mathbb{N}$, $X_{1..N} \in \text{String}^N$

Output: longest $\in \text{String}_1^N$

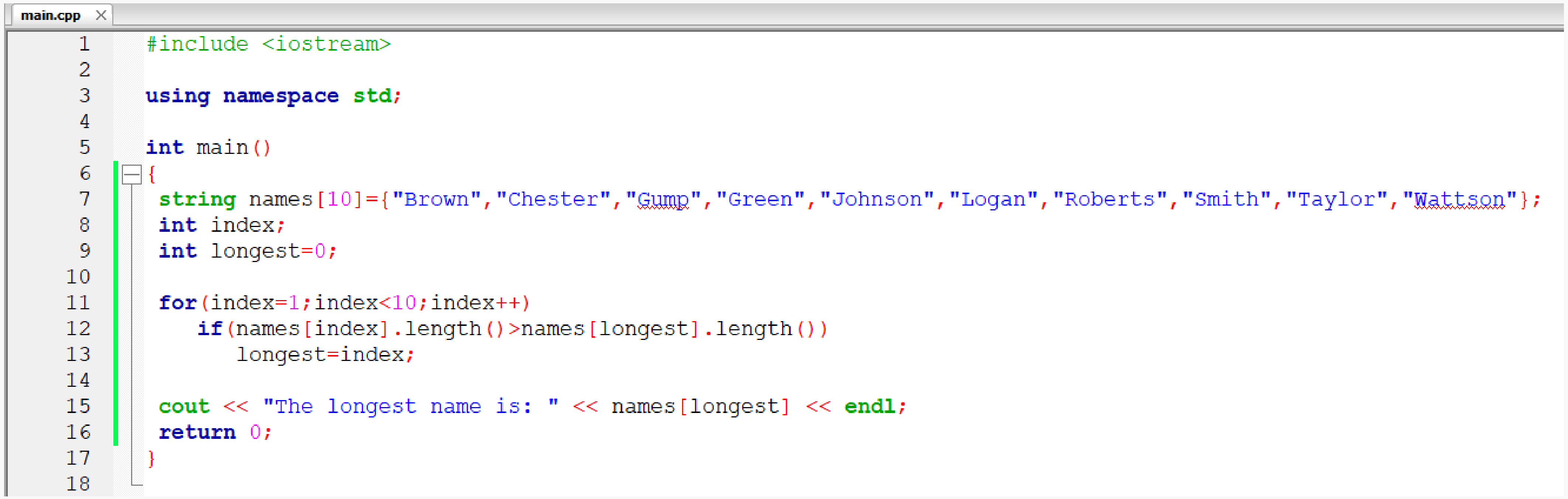
Precondition: $N > 0$ and $\forall i\ (1 \leq i \leq N)$: $\text{length}(X_i) > 0$

Postcondition: $\forall i\ (1 \leq i \leq N)$: $\text{length}(X_{max}) \geq \text{length}(X_i)$ and longest $= X_{max}$


The Longest Name

```
longest:=X[1]
i:= 2..N
    length(X[i])>length(longest)
    T                          F
    longest:=X[i]              ∅
```

Source code 4 – Maximum selection – The longest name



main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
 string names[10]={"Brown","Chester","Gump","Green","Johnson","Logan","Roberts","Smith","Taylor","Wattson"};
 int index;
 int longest=0;

 for(index=1;index<10;index++)
    if(names[index].length()>names[longest].length())
       longest=index;

 cout << "The longest name is: " << names[longest] << endl;
 return 0;
}

```

Example 5 – Search

Square number: Let's looking for a square number within a sequence of numbers.

Specification:

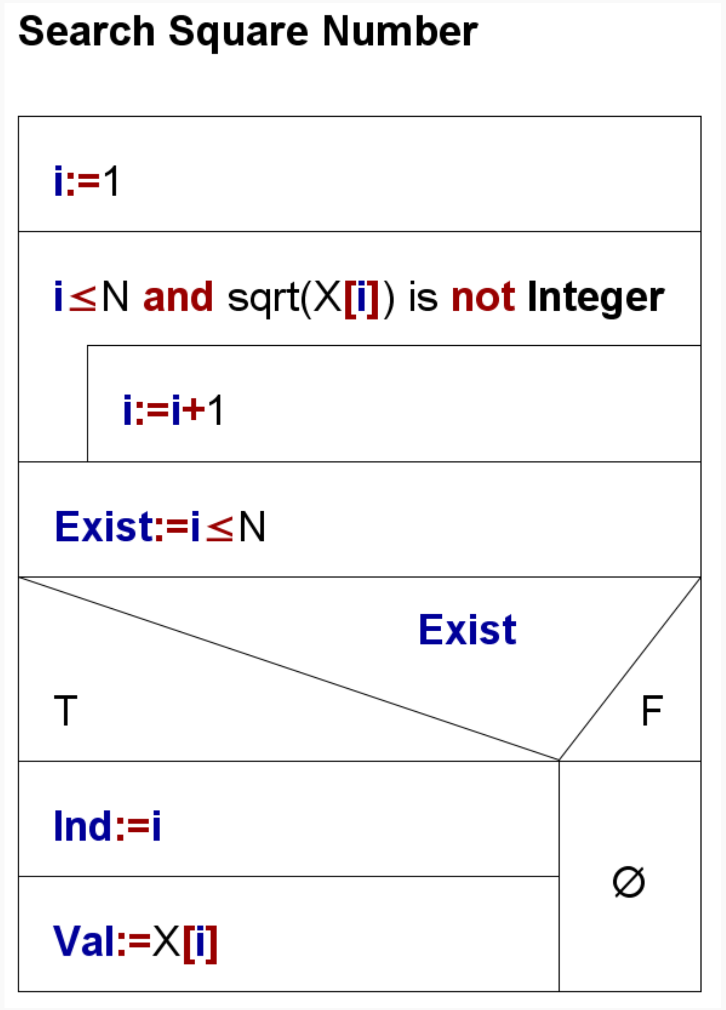
Input: $N \in \mathbb{N}$, $X_{1..N} \in \mathbb{N}^N$, $A: \mathbb{N} \rightarrow L$

Output: $Exists \in L$, $Ind \in \mathbb{N}$, $Val \in \mathbb{N}$

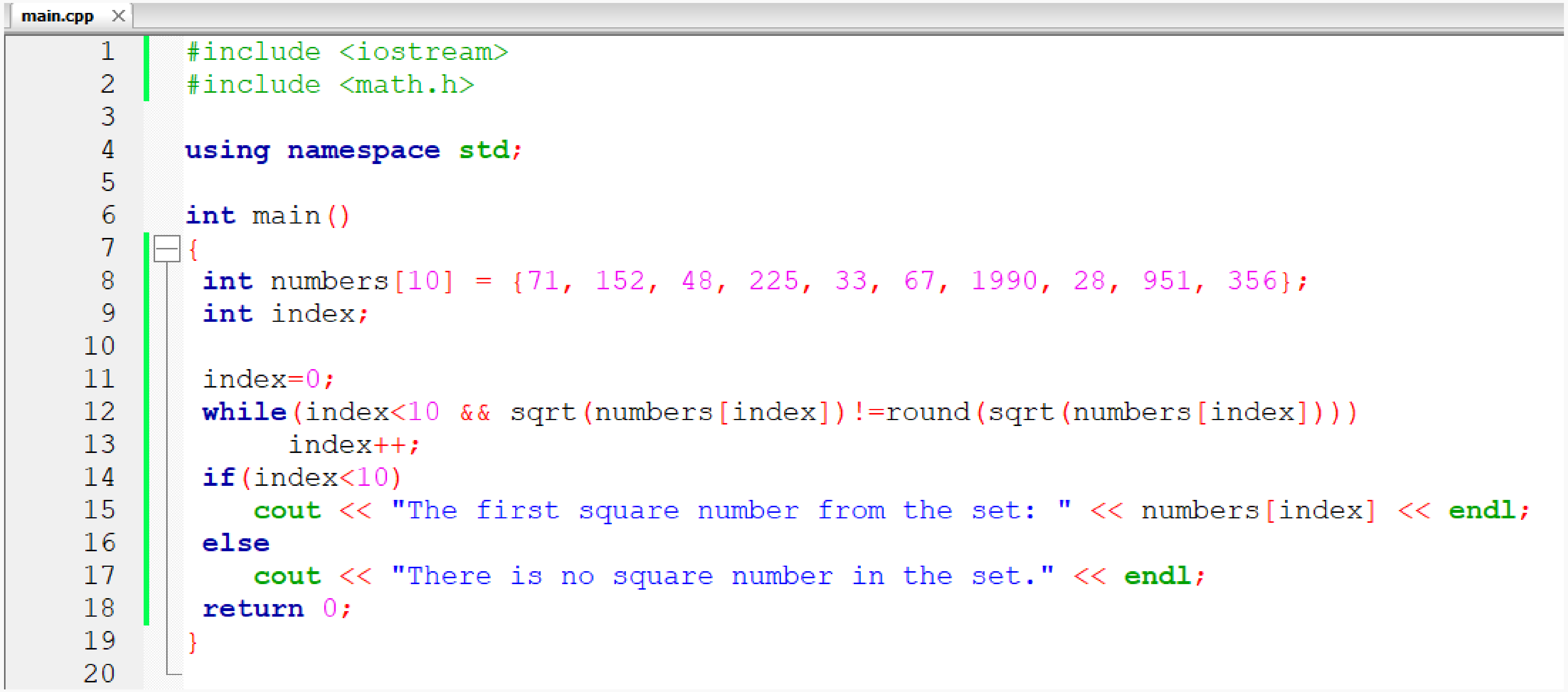
Precondition: $\forall\, i\ (1 \leq i \leq N) : X_i \geq 0$

Postcondition: $Exists = (\exists\, i\ (1 \leq i \leq N) : sqrt(X_i)$ is Integer

and $1 \leq Ind \leq N$ and $sqrt(X_{Ind})$ is Integer



Source code 5 – Search – Square number



```cpp
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
 int numbers[10] = {71, 152, 48, 225, 33, 67, 1990, 28, 951, 356};
 int index;

 index=0;
 while(index<10 && sqrt(numbers[index])!=round(sqrt(numbers[index])))
      index++;
 if(index<10)
    cout << "The first square number from the set: " << numbers[index] << endl;
 else
    cout << "There is no square number in the set." << endl;
 return 0;
}
```

Example 6 – Decision

Divisible by 3: Let's make a decision is there any number that is divisible by 3 within a sequence of numbers.

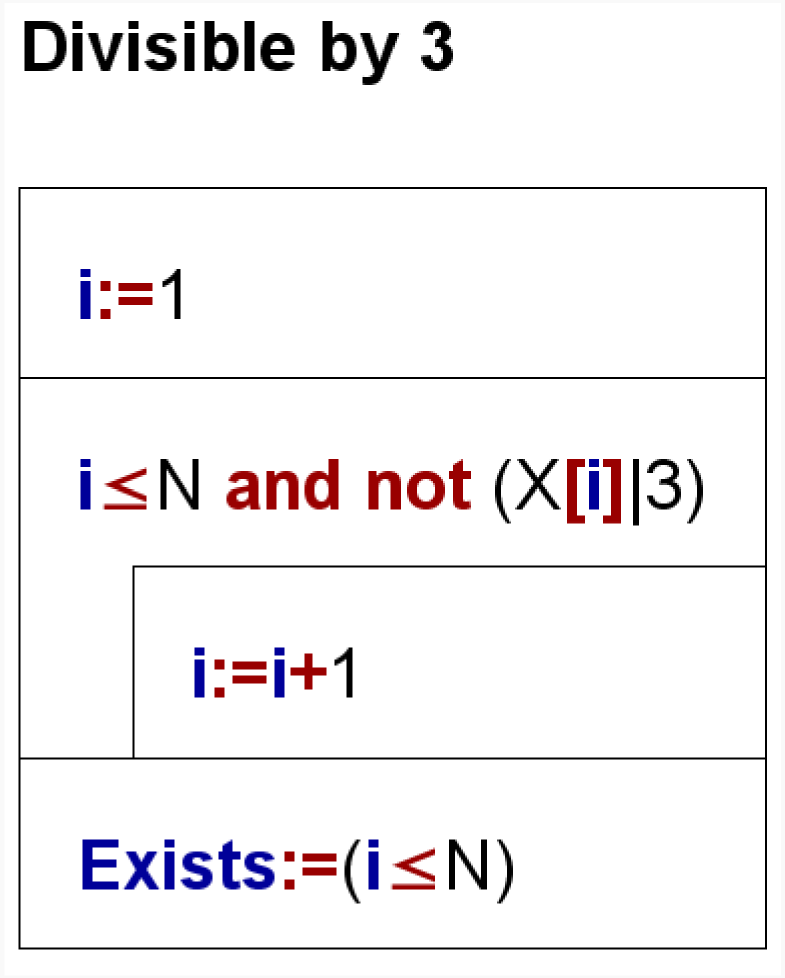
Specification:

Input: $N \in \mathbb{N}$, $X_{1..N} \in \mathbb{N}^N$, $A: \mathbb{N} \rightarrow L$

Output: Exists $\in L$

Precondition: –

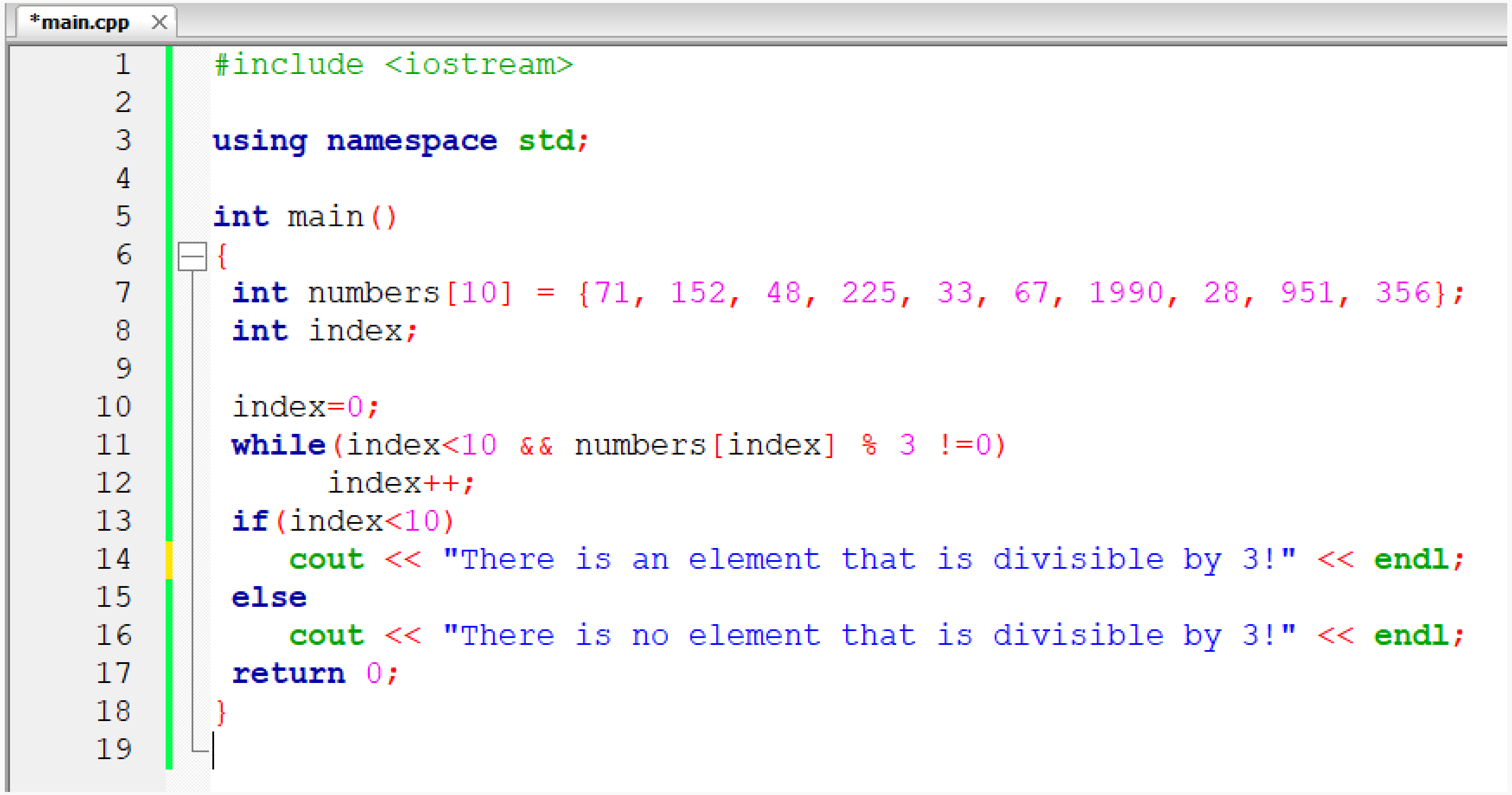
Postcondition: Exists = $\exists\, i\ (1 \leq i \leq N): X_i | 3$



Divisible by 3

```
i:=1
i≤N and not (X[i]|3)
    i:=i+1
Exists:=(i≤N)
```

Source code 6 – Decision – Divisible by 3



```cpp
#include <iostream>

using namespace std;

int main()
{
 int numbers[10] = {71, 152, 48, 225, 33, 67, 1990, 28, 951, 356};
 int index;

 index=0;
 while(index<10 && numbers[index] % 3 !=0)
      index++;
 if(index<10)
    cout << "There is an element that is divisible by 3!" << endl;
 else
    cout << "There is no element that is divisible by 3!" << endl;
 return 0;
}

```

Example 7 – Selection

Four digits number: Let's select the first four digits number from a sequence of numbers.

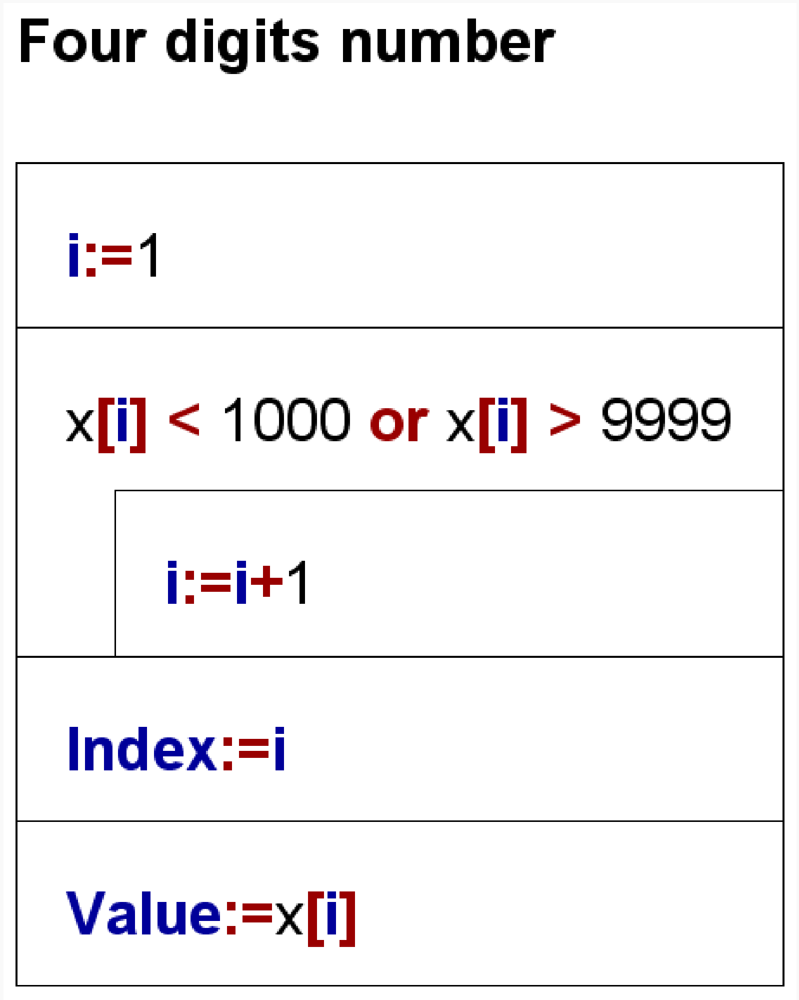
Specification:

Input: $N \in \mathbb{N}$, $X_{1..N} \in \mathbb{N}^N$, $A: \mathbb{N} \rightarrow L$

Output: $Index \in \mathbb{N}$, $Value \in \mathbb{N}$

Precondition: $\mathbb{N} > 0$ and $\exists\, i\ (1 \leq i \leq N){:}A(X_i)$

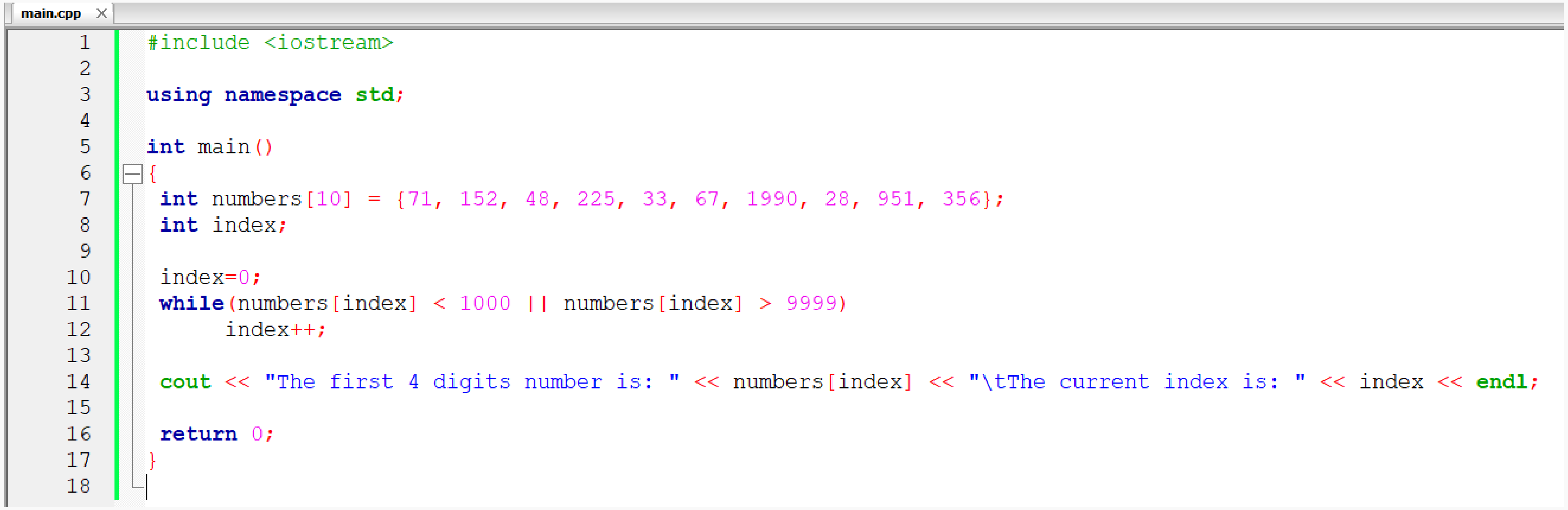
Postcondition: $1 \leq Index \leq N$ and $1000 \leq X_{index} \leq 9999$



Four digits number

```
i:=1
x[i] < 1000 or x[i] > 9999
    i:=i+1
Index:=i
Value:=x[i]
```

Source code 7 – Selection – Four digits number


main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
 int numbers[10] = {71, 152, 48, 225, 33, 67, 1990, 28, 951, 356};
 int index;

 index=0;
 while(numbers[index] < 1000 || numbers[index] > 9999)
      index++;

 cout << "The first 4 digits number is: " << numbers[index] << "\tThe current index is: " << index << endl;

 return 0;
}

```