SarFTI NRNU MEPhI
Faculty of Information Technology and Electronics
Department of Computing and Information Technology

THINKING FUNCTIONALLY WITH HASKELL

COMPLETED BY A STUDENT OF GROUP VTM-10

VLADISLAV ZLOBIN

Sarov 2021

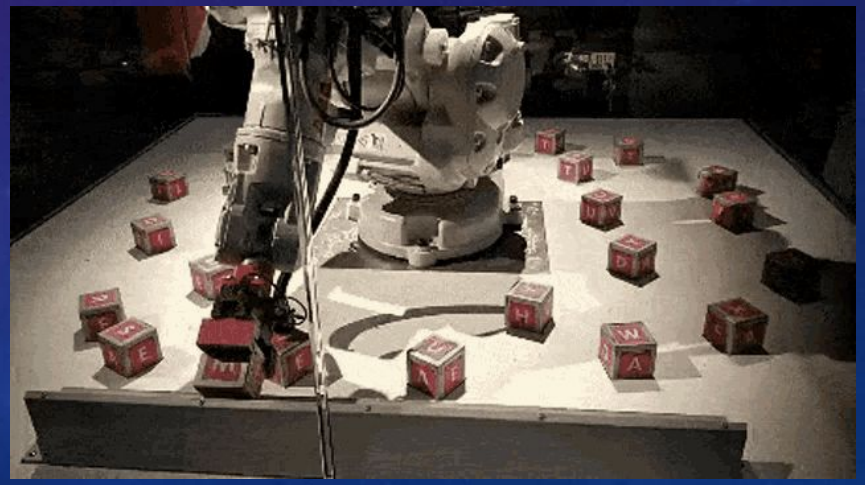
TYPES OF PERFORMERS

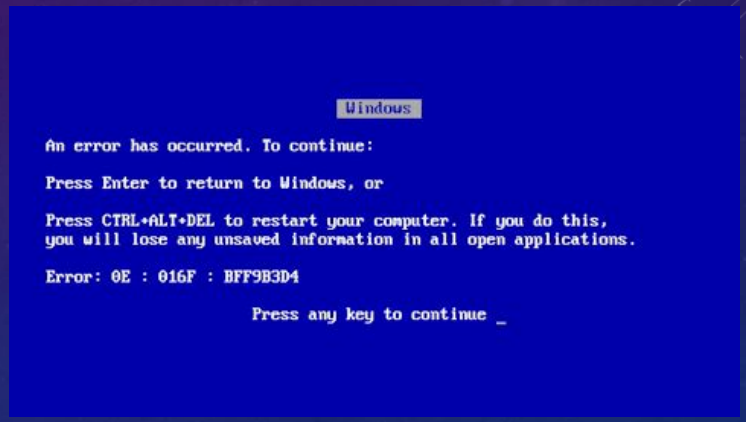
Informal

Can execute the same algorithm in different ways.

formal

Performs the same algorithm the same way.




Windows
An error has occurred. To continue:
Press Enter to return to Windows, or
Press CTRL+ALT+DEL to restart your computer. If you do this,
you will lose any unsaved information in all open applications.
Error: 0E : 016F : BFF9B3D4
Press any key to continue _

PROGRAMMING

PROGRAMMING PARADIGMS

Imperative (procedural, structural, object-oriented, etc.)

Declarative (functional and logical)

FUNCTIONAL PROGRAMMING

The basis of functional programming is the calculation of functions (in mathematical terms).

$$f(x) = 2x + 10$$

```
ghci> f x = 2*x + 10
ghci> f 5
20
```

PURE FUNCTION

In programming languages, a pure function is a function that:

- Is deterministic;
- Has no side effects.

THE DIFFERENCE BETWEEN IMPERATIVE AND FUNCTIONAL PL

The main difference is that imperative languages have states, functional languages do not.

Debugging state changes in imperative languages

No need to debug states in functional languages

Is pure functional programming language

Because:

- It supports pure functions only;
- It has no states.

LAZY EVALUATION

Lazy computation allows Haskell to reduce the total amount of computation at the expense of computations that will not be used. The programmer can simply describe the dependencies of functions from each other and not make sure that "unnecessary calculations" are not carried out.

STATIC TYPING

Haskell has static strong full typing with automatic type inference. These measures allow you to effectively catch bugs at the stage of compiling the source code.

AN EXAMPLE OF SOLVING A TASK IN HASKELL

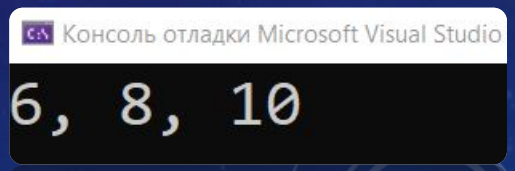
Find all possible right-angled triangles whose side lengths are in the range from 1 to 10 (natural numbers) and whose perimeter is 24.


```
ghci> rightTriangles = [ (a,b,c) | c <- [1..10], b <- [1..c], a <- [1..b], a^2 + b^2 == c^2, a+b+c == 24]
ghci> rightTriangles
[(6,8,10)]
```

COMPARISON WITH OTHER PL


```
>>> rightTriangles = [(a, b, c) for c in range(1, 11) for b in range(1, c + 1) for a in range(1, b + 1) if a**2 + b**2 == c**2 if a + b + c == 24]
>>> rightTriangles
[(6, 8, 10)]
```

```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <string>

using namespace std;

int main()
{
    vector<vector<int>> container;
    for (int c = 1; c <= 10; c++)
        for (int b = 1; b <= c; b++)
            for (int a = 1; a <= b; a++)
                if (pow(a, 2) + pow(b, 2) == pow(c, 2) && a + b + c == 24)
                    container.push_back(vector<int>{a, b, c});

    for (int i = 0; i < container.size(); i++)
        cout << container.at(i)[0] << ", "
             << container.at(i)[1] << ", "
             << container.at(i)[2] << endl;

    return 0;
}

```



Thanks for attention!