

# Задание 1

В первом задании, чтобы запускать c++ код на c# & java, мы конвертировали c++ код в dll библиотеки, и использовали их на соответствующих языках

Для C#:

Для этого в cpp файле прописываем саму функцию

```
#include "library.h"

int sum(int a, int b) {
    return a + b;
}
```

А в хедере описываем, то что будем её конвертировать в dll библиотеку

```
#pragma once
#ifdef MATH_API
# define MATH_API __declspec(dllexport)
#else
# define MATH_API __declspec(dllimport)
#endif

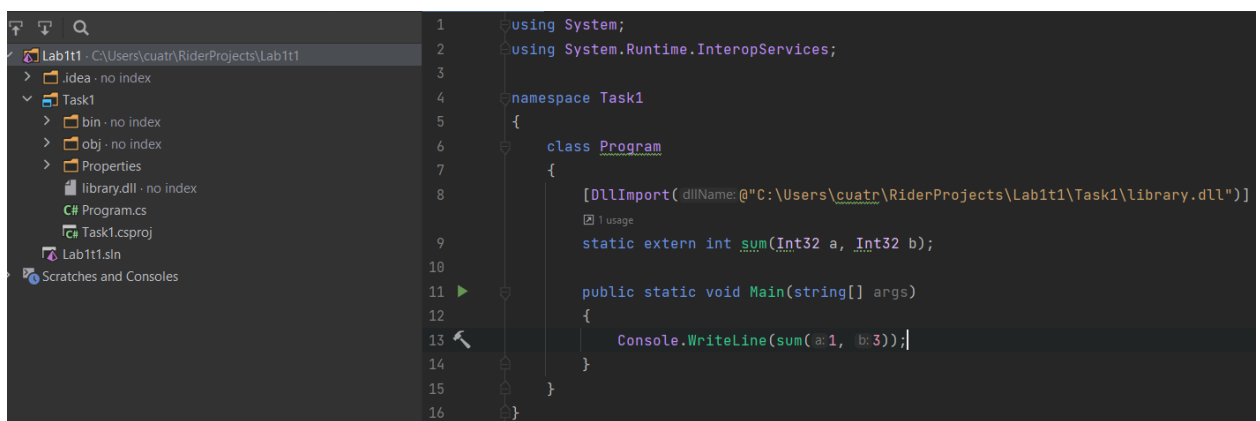
extern "C" int sum(int, int);
```

Затем, чтобы создать dll прописываем в консоли следующие команды:

```
g++ -c -m32 library.cpp
```

```
g++ -shared -m32 -o library.dll library.o
```

примечательно то, что пока не появилось -m32, оно наотказ отказывалось работать в шарпах ссылаясь на неподходящий формат, затем для надежности переносим полученную библиотеку в шарповый проэктик:



```
1 using System;
2 using System.Runtime.InteropServices;
3
4 namespace Task1
5 {
6     class Program
7     {
8         [DllImport("C:\\Users\\cuatr\\RiderProjects\\Lab1t1\\Task1\\library.dll")]
9         static extern int sum(Int32 a, Int32 b);
10
11         public static void Main(string[] args)
12         {
13             Console.WriteLine(sum(1, 3));
14         }
15     }
16 }
```

В нем мы прописываем импорт библиотеки и путь до нее, теперь мы можем использовать функцию суммирования, перед этим описав её заголовок

Для джавы:


Код сpp немного меняется:

```
#include "library.h"

JNIEXPORT jint JNICALL Java_Main_sum
(JNIEnv* env, jobject thisObject, jint a, jint b) {
    return a + b;
}
```

Мы говорим, что переменные и прочее будут специально для джавы, хедер мы тоже морально подготавливаем к тому, что код будет использоваться в джаве:

```
#include <jni.h>

#ifndef add
#define add
#ifdef __cplusplus
extern "C" {
#endif

JNIEXPORT jint JNICALL Java_Main_sum
(JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}
#endif
#endif
```

На этот раз пишем другую команду в консоли для dll для джавы:

```
g++ -I "C:\Users\cuatr\jdk\openjdk-17.0.2\include" -I "C:\Users\cuatr\jdk\openjdk-17.0.2\include\win32" -shared -o library.dll library.cpp
```

где указываем путь до jdk и опять полученную библиотеку переносим на этот раз в джава проект:

```

public class Main {
    static {
        System.load( filename: "C:\\Users\\cuatr\\IdeaProjects\\L1Task1\\src\\library.dll");
    }

    public static void main(String[] args) {
        System.out.println("sddfqfd");
        System.out.println(new Main().sum( a: 1, b: 3));
    }

    public native int sum(int a, int b);
}

```

В нем, собственно, успешно запускаем и выполняем нашу сумму

## Задание 2

Начнем с фарша:

Для начала пайп:

```

namespace fs

module pipe =
    int -> int
    let f1 x =
        x * x
    int -> int
    let f2 x =
        x * 2
    int -> int
    let start x =
        x |> f1 |> f2

```

Он позволяет перенаправлять выходящие значения на вход для следующих функций

Если посмотреть на декомпиляцию кода в шарпе:

```

using System.Reflection;
using Microsoft.FSharp.Core;

[assembly: FSharpInterfaceDataVersion(2, 0, 0)]
[assembly: AssemblyVersion("0.0.0.0")]
namespace fs
{
    [CompilationMapping(SourceConstructFlags.Module)]
    public static class pipe
    {
        public static int f1(int x)
        {
            return x * x;
        }

        public static int f2(int x)
        {
            return x * 2;
        }

        public static int start(int x)
        {
            return x * x * 2;
        }
    }
}
namespace <StartupCode$_>
{
    internal static class $_
    {
    }
}

```

Видим, что в старт просто были перенесены все операции с переменной из предыдущих функций, Discriminated Union, грубо говоря компактно описанный класс с классами

```

module DiscUnion =
  type Shape =
    | Rectangle of width : float * length : float
    | Circle of radius : float
    | Prism of width : float * float * height : float

```

Убедимся в этом в декомпилированном коде:

<https://sharplab.io/#v2:DYLgZgzgNAJiDUAfAdgQwLYFMIAdUGNMACSAWACgLOB7GAV2GIBELCfAVWReuSIF4KRYUQAuATxzEAYgAtUUgUJGIIAJUz5RqZAHNGRamCIB3FjFGyilEsGqpRRAFRFGey9dv3Ry4aoDCLABO+AZGREGoMCx0EJ5gdg6+RkoACkFs6lbGZhZWNgnetzl4OxbKYLLqyjgWJPuQijUTJjUA===>

ну там получается какая-то реклама фаршей, потому что аналог на шарпах занимает несколько метров экрана, но там можно различить класс Shape и его подклассы Rectangle, и т.д.

Computation expressions, или вычислительные выражения в фарше предоставляют удобный синтаксис для написания вычислений, которые могут быть упорядочены и объединены с помощью конструкций и привязок потока управления. В зависимости от типа вычислительного выражения их можно представить как способ выражения составных, моноидс, нестандартных преобразователей и аппликативе операторов. Однако, в отличие от других языков (например, в Haskell), они не привязаны к одной абстракции и не полагаются на макросы или другие формы метапрограммирования для выполнения удобного и контекстно-чувствительного синтаксиса.

```

module ComputationEx =
  (int * int * int * int * int * int * int * int * int * int) list
  let list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  seq<int>
  let square =
    seq {
      for i in 1..10 do
        yield i * i
        printf $"%A{i}"
    }

```

В шарпах это:

<https://sharplab.io/#v2:DYLgZgzgNAJiDUAfAdgQwLYFMIAdUGNMACSAWACgLOB7GAV2GIGFr0c6AXVDgS2uQCiADylBeCkUIFGHaTwizRRANoAGKEQCMGgEwaAzBoAsGgKwaAbBoDsGgBwaAnAF0JUUmUQgBHOqgBOxOLkUiGemF5EAN5uoSTUfkQ8ichaAHSpmqpEMNQxsVIANjyYwDCJRABUiXn5RDh+PMgcYEQAJABEAKQAgpE8AL7tNVL9w5JjREA===>

тут уже не так страшно как в предыдущем случае, но все равно много, и само собой мы можем вызывать эти функции из шарпов просто добавив референс между проектами:

```
using System;

namespace MyNamespace
{
    class MyClass
    {
        public static void Main(string[] args)
        {
            Console.WriteLine(fs.pipe.start(x: 11));
            var shape = fs.DiscUnion.Shape.NewCircle(10);
            Console.WriteLine(shape.IsPrism);
            Console.WriteLine(fs.ComputationEx.list);
        }
    }
}
```

Для скалы и джавы соответственно будет что-то подобное, только со своими проблемами

```

package main

import scala.concurrent.Future
import scala.util.chaining._

object pipe {
  def h(x: Int) = x + x
  def i(x: Int) = x * x
  def pip(x: Int) = x.pipe(h).pipe(i)

  sealed trait Shape
  case class Circle(r: Double) extends Shape
  case class Square(s: Double) extends Shape
  case class Rectangle(h: Double, w: Double) extends Shape

  def area(shape: Shape) : Double = shape match {
    case Circle(r) => Math.PI * r * r
    case Square(s) => s * s
    case Rectangle(h, w) => h * w
  }
}

```

И подвязав джаву можем из неё безболезненно исполнить написанные на скале функции

```

package main;

public class T2 {
    public static void main(String[] args) {
        System.out.println(pipe.pip(3));
        System.out.println(pipe.area(new pipe.Rectangle(5,4)));
    }
}

```

Ну и само собой 500 строк того же кода только декомпилированного в джаву:

```

//decompiled from pipe.class
package main;

import java.io.Serializable;
import scala.Product;
import scala.collection.Iterator;

```





```

        var10000 = this.r() == var3.r() && var3.canEqual(this);
    } else {
        var10000 = false;
    }

    if (!var10000) {
        var10000 = false;
        return var10000;
    }
}

var10000 = true;
return var10000;
}

public String toString() {
    return scala.runtime.ScalaRunTime..MODULE$.toString(this);
}

public boolean canEqual(final Object that) {
    return that instanceof pipe.Circle;
}

public int productArity() {
    return 1;
}

public String productPrefix() {
    return "Circle";
}

public Object productElement(final int n) {
    if (0 == n) {
        return BoxesRunTime.boxToDouble(this._1());
    } else {
        throw new
IndexOutOfBoundsException(BoxesRunTime.boxToInteger(n).toString());
    }
}

public String productElementName(final int n) {
    if (0 == n) {
        return "r";
    } else {
        throw new
IndexOutOfBoundsException(BoxesRunTime.boxToInteger(n).toString());
    }
}

public double r() {
    return this.r;
}

public pipe.Circle copy(final double r) {
    return new pipe.Circle(r);
}

public double copy$default$1() {
    return this.r();
}

public double _1() {
    return this.r();
}

```

```

    }

    public static final class Circle$ implements scala.deriving.Mirror.Product,
    Serializable {
        public static final pipe.Circle$ MODULE$ = new pipe.Circle$();

        private Object writeReplace() {
            return new ModuleSerializationProxy(pipe.Circle$.class);
        }

        public pipe.Circle apply(final double r) {
            return new pipe.Circle(r);
        }

        public pipe.Circle unapply(final pipe.Circle x$1) {
            return x$1;
        }

        public String toString() {
            return "Circle";
        }

        public pipe.Circle fromProduct(final Product x$0) {
            return new
pipe.Circle(BoxesRunTime.unboxToDouble(x$0.productElement(0)));
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Object fromProduct(final Product p) {
            return this.fromProduct(p);
        }
    }

    public static class Rectangle implements pipe.Shape, Product, Serializable {
        private final double h;
        private final double w;

        public static pipe.Rectangle apply(double var0, double var2) {
            return pipe.Rectangle$.MODULE$.apply(var0, var2);
        }

        public static pipe.Rectangle fromProduct(Product var0) {
            return pipe.Rectangle$.MODULE$.fromProduct(var0);
        }

        public static pipe.Rectangle unapply(pipe.Rectangle var0) {
            return pipe.Rectangle$.MODULE$.unapply(var0);
        }

        public Rectangle(final double h, final double w) {
            this.h = h;
            this.w = w;
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Iterator productIterator() {
            return Product.productIterator$(this);
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Iterator productElementNames() {

```

```

        return Product.productElementNames$(this);
    }

    public int hashCode() {
        int var1 = -889275714;
        var1 = Statics.mix(var1, this.productPrefix().hashCode());
        var1 = Statics.mix(var1, Statics.doubleHash(this.h()));
        var1 = Statics.mix(var1, Statics.doubleHash(this.w()));
        return Statics.finalizeHash(var1, 2);
    }

    public boolean equals(final Object x$0) {
        boolean var10000;
        if (this != x$0) {
            if (x$0 instanceof pipe.Rectangle) {
                pipe.Rectangle var3 = (pipe.Rectangle)x$0;
                var10000 = this.h() == var3.h() && this.w() == var3.w() &&
var3.canEqual(this);
            } else {
                var10000 = false;
            }

            if (!var10000) {
                var10000 = false;
                return var10000;
            }
        }

        var10000 = true;
        return var10000;
    }

    public String toString() {
        return scala.runtime.ScalaRunTime..MODULE$.toString(this);
    }

    public boolean canEqual(final Object that) {
        return that instanceof pipe.Rectangle;
    }

    public int productArity() {
        return 2;
    }

    public String productPrefix() {
        return "Rectangle";
    }

    public Object productElement(final int n) {
        double var10000;
        if (0 == n) {
            var10000 = this._1();
        } else {
            if (1 != n) {
                throw new
IndexOutOfBoundsException(BoxesRunTime.boxToInteger(n).toString());
            }

            var10000 = this._2();
        }

        return BoxesRunTime.boxToDouble(var10000);
    }
}

```

```

    public String productElementName(final int n) {
        String var10000;
        if (0 == n) {
            var10000 = "h";
        } else {
            if (1 != n) {
                throw new
IndexOutOfBoundsException(BoxesRunTime.boxToInteger(n).toString());
            }

            var10000 = "w";
        }

        return var10000;
    }

    public double h() {
        return this.h;
    }

    public double w() {
        return this.w;
    }

    public pipe.Rectangle copy(final double h, final double w) {
        return new pipe.Rectangle(h, w);
    }

    public double copy$default$1() {
        return this.h();
    }

    public double copy$default$2() {
        return this.w();
    }

    public double _1() {
        return this.h();
    }

    public double _2() {
        return this.w();
    }
}

public static final class Rectangle$ implements
scala.deriving.Mirror.Product, Serializable {
    public static final pipe.Rectangle$ MODULE$ = new pipe.Rectangle$();

    private Object writeReplace() {
        return new ModuleSerializationProxy(pipe.Rectangle$.class);
    }

    public pipe.Rectangle apply(final double h, final double w) {
        return new pipe.Rectangle(h, w);
    }

    public pipe.Rectangle unapply(final pipe.Rectangle x$1) {
        return x$1;
    }

    public String toString() {
        return "Rectangle";
    }
}

```

```

        public pipe.Rectangle fromProduct(final Product x$0) {
            return new
pipe.Rectangle(BoxesRunTime.unboxToDouble(x$0.productElement(0)),
BoxesRunTime.unboxToDouble(x$0.productElement(1)));
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Object fromProduct(final Product p) {
            return this.fromProduct(p);
        }
    }

    public interface Shape {
    }

    public static class Square implements pipe.Shape, Product, Serializable {
        private final double s;

        public static pipe.Square apply(double var0) {
            return pipe.Square$.MODULE$.apply(var0);
        }

        public static pipe.Square fromProduct(Product var0) {
            return pipe.Square$.MODULE$.fromProduct(var0);
        }

        public static pipe.Square unapply(pipe.Square var0) {
            return pipe.Square$.MODULE$.unapply(var0);
        }

        public Square(final double s) {
            this.s = s;
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Iterator productIterator() {
            return Product.productIterator$(this);
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Iterator productElementNames() {
            return Product.productElementNames$(this);
        }

        public int hashCode() {
            int var1 = -889275714;
            var1 = Statics.mix(var1, this.productPrefix().hashCode());
            var1 = Statics.mix(var1, Statics.doubleHash(this.s()));
            return Statics.finalizeHash(var1, 1);
        }

        public boolean equals(final Object x$0) {
            boolean var10000;
            if (this != x$0) {
                if (x$0 instanceof pipe.Square) {
                    pipe.Square var3 = (pipe.Square)x$0;
                    var10000 = this.s() == var3.s() && var3.canEqual(this);
                } else {
                    var10000 = false;
                }
            }
        }
    }

```

```

        if (!var10000) {
            var10000 = false;
            return var10000;
        }

        var10000 = true;
        return var10000;
    }

    public String toString() {
        return scala.runtime.ScalaRunTime..MODULE$._toString(this);
    }

    public boolean canEqual(final Object that) {
        return that instanceof pipe.Square;
    }

    public int productArity() {
        return 1;
    }

    public String productPrefix() {
        return "Square";
    }

    public Object productElement(final int n) {
        if (0 == n) {
            return BoxesRunTime.boxToDouble(this._1());
        } else {
            throw new
IndexOutOfBoundsException(BoxesRunTime.boxToInteger(n).toString());
        }
    }

    public String productElementName(final int n) {
        if (0 == n) {
            return "s";
        } else {
            throw new
IndexOutOfBoundsException(BoxesRunTime.boxToInteger(n).toString());
        }
    }

    public double s() {
        return this.s;
    }

    public pipe.Square copy(final double s) {
        return new pipe.Square(s);
    }

    public double copy$default$1() {
        return this.s();
    }

    public double _1() {
        return this.s();
    }
}

public static final class Square$ implements scala.deriving.Mirror.Product,
Serializable {

```

```

        public static final pipe.Square$ MODULE$ = new pipe.Square$();

        private Object writeReplace() {
            return new ModuleSerializationProxy(pipe.Square$.class);
        }

        public pipe.Square apply(final double s) {
            return new pipe.Square(s);
        }

        public pipe.Square unapply(final pipe.Square x$l) {
            return x$l;
        }

        public String toString() {
            return "Square";
        }

        public pipe.Square fromProduct(final Product x$0) {
            return new
pipe.Square(BoxesRunTime.unboxToDouble(x$0.productElement(0)));
        }

        // $FF: synthetic method
        // $FF: bridge method
        public Object fromProduct(final Product p) {
            return this.fromProduct(p);
        }
    }
}

//decompiled from pipe$.class
package main;

import java.io.Serializable;
import java.lang.invoke.SerializedLambda;
import scala.MatchError;
import scala.runtime.BoxesRunTime;
import scala.runtime.ModuleSerializationProxy;
import scala.util.package.chaining.;

public final class pipe$ implements Serializable {
    public static final pipe.Circle$ Circle;
    public static final pipe.Square$ Square;
    public static final pipe.Rectangle$ Rectangle;
    public static final pipe$ MODULE$ = new pipe$();

    private pipe$() {
    }

    private Object writeReplace() {
        return new ModuleSerializationProxy(pipe$.class);
    }

    public int h(final int x) {
        return x + x;
    }

    public int i(final int x) {
        return x * x;
    }

    public int pip(final int x) {
        var10000 = .MODULE$;

```

```

        Integer var3 =
(Integer).MODULE$.scalaUtilChainingOps (BoxesRunTime.boxToInteger (x) );
        Integer var2 =
(Integer)var10000.scalaUtilChainingOps (scala.util.ChainingOps..MODULE$.pipe$exte
nsion(var3, (xx) -> {
            return this.h(xx);
        }));
        return
BoxesRunTime.unboxToInt (scala.util.ChainingOps..MODULE$.pipe$extension (var2,
(xx) -> {
            return this.i(xx);
        }));
    }

    public double area(final pipe.Shape shape) {
        double var10000;
        if (shape instanceof pipe.Circle) {
            pipe.Circle var3 = pipe.Circle$.MODULE$.unapply((pipe.Circle) shape);
            double var4 = var3._1();
            var10000 = 3.141592653589793D * var4 * var4;
        } else if (shape instanceof pipe.Square) {
            pipe.Square var8 = pipe.Square$.MODULE$.unapply((pipe.Square) shape);
            double var9 = var8._1();
            var10000 = var9 * var9;
        } else {
            if (!(shape instanceof pipe.Rectangle)) {
                throw new MatchError(shape);
            }

            pipe.Rectangle var13 =
pipe.Rectangle$.MODULE$.unapply((pipe.Rectangle) shape);
            double var14 = var13._1();
            double var16 = var13._2();
            var10000 = var14 * var16;
        }

        return var10000;
    }

    // $FF: synthetic method
    private static Object $deserializeLambda$(SerializedLambda var0) {
        return var0.lambdaDeserialize<invokedynamic>(var0);
    }
}

```

## Задание 3

Для начала пишем (берем из интернетов) ДФС, БФС

```

import java.util.*;

class Graph {
    private int V; // No. of vertices

    // Array of lists for
    // Adjacency List Representation
    private LinkedList<Integer> adj[];

    // Constructor
    @SuppressWarnings("unchecked") Graph(int v)

```



```

{
    V = v;
    adj = new LinkedList[v];
    for (int i = 0; i < v; ++i)
        adj[i] = new LinkedList();
}

// Function to add an edge into the graph
void addEdge(int v, int w)
{
    adj[v].add(w); // Add w to v's list.
}

// A function used by DFS
void DFSUtil(int v, boolean visited[])
{
    // Mark the current node as visited and print it
    visited[v] = true;
    System.out.print(v + " ");

    // Recur for all the vertices adjacent to this
    // vertex
    Iterator<Integer> i = adj[v].listIterator();
    while (i.hasNext()) {
        int n = i.next();
        if (!visited[n])
            DFSUtil(n, visited);
    }
}

// The function to do DFS traversal.
// It uses recursive
// DFSUtil()
void DFS(int v)
{
    // Mark all the vertices as
    // not visited(set as
    // false by default in java)
    boolean visited[] = new boolean[V];

    // Call the recursive helper
    // function to print DFS
    // traversal
    DFSUtil(v, visited);
}

void BFS(int s)
{
    // Mark all the vertices as not visited(By default
    // set as false)
    boolean visited[] = new boolean[V];

    // Create a queue for BFS
    LinkedList<Integer> queue = new LinkedList<Integer>();

    // Mark the current node as visited and enqueue it
    visited[s]=true;
    queue.add(s);

    while (queue.size() != 0)
    {
        // Dequeue a vertex from queue and print it
        s = queue.poll();
        System.out.print(s+" ");
    }
}

```

```

        // Get all adjacent vertices of the dequeued vertex s
        // If a adjacent has not been visited, then mark it
        // visited and enqueue it
        Iterator<Integer> i = adj[s].listIterator();
        while (i.hasNext())
        {
            int n = i.next();
            if (!visited[n])
            {
                visited[n] = true;
                queue.add(n);
            }
        }
    }
}

// Driver Code
public static void main(String args[])
{
    Graph g = new Graph(4);

    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    System.out.println(
        "Following is Depth First Traversal "
        + "(starting from vertex 2)");

    g.DFS(2);
    System.out.println();
    g.BFS(0);
}
}
// This code is contributed by Aakash Hasija

```

теперь через артефакты превращаем наш написанный код в формат jar, который уже можно опубликовать, но мы не настолько амбициозны, поэтому просто скопируем этот файл в проект со скалой

```

object Main {
  def main(args: Array[String]): Unit = {
    val g = new Graph(4)
    g.addEdge(0, 1)
    g.addEdge(0, 2)
    g.addEdge(1, 2)
    g.addEdge(2, 0)
    g.addEdge(2, 3)
    g.addEdge(3, 3)

    g.DFS(2)
    g.BFS(0);
  }
}

```

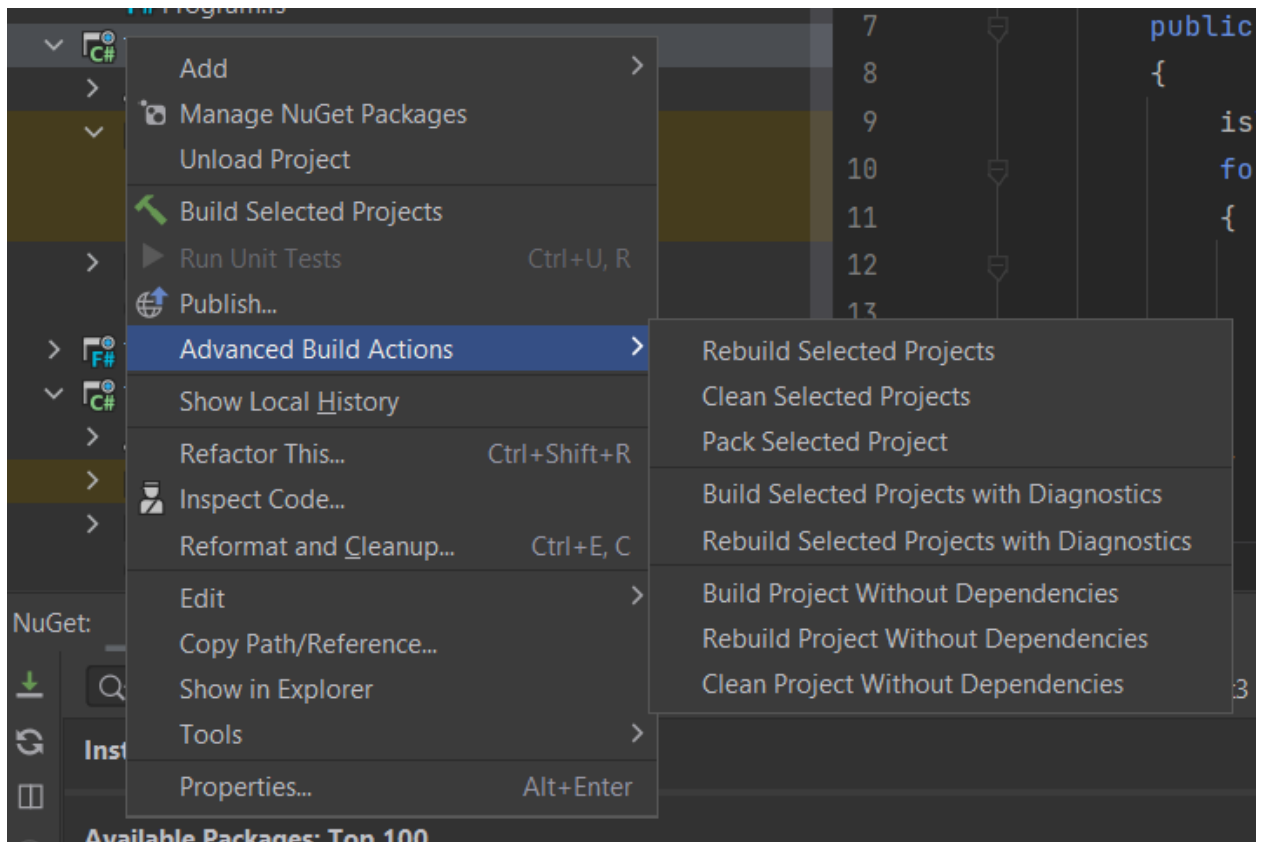
где его можно спокойно запустить, при условии, что мы создавали скалу через мавен, и соответственно с помощью него подключаем наш jar

```

<orderEntry type="module-library" exported="">
  <library>
    <CLASSES>
      <root url="jar://$MODULE_DIR$/lib/Lab1.jar!/" />
    </CLASSES>
    <JAVADOC />
    <SOURCES />
  </library>
</orderEntry>
<orderEntry type="library" name="scala-sdk-3.1.1" level="application" />

```

В шарпах в этом плане все достаточно приятнее



Берем написанный класс (в этот раз самостоятельно, поэтому криво) пакуем его и можно смело публиковать, затем просто скачиваем в фарш и все работает:

```

open TaskDDI3

printfn "Hello from F#"
obj
let a = Class1()

bool[,]
let myGraph =
    array2D [ [ true; false; true ]
              [ false; false; true ]
              [ false; true; false ] ]

obj
let visit = a.Bfs(myGraph, 0)
printfn $"%A{visit}"

```

## Задание 4

В этот раз начнем с шарпов, тут в классе с сортировками перед функциями, представляющие наши сортировки, в квадратных скобках пишем бенчмарк, не забываем само собой установить его перед этим в нугете, тогда описав сам мейн примерно так:

```

public class Program
{
    public static void Main(string[] args)
    {
        var summary = BenchmarkRunner.Run<Sorts>();
    }
}

```

Мы запустим тесты где он по несколько раз запускает код чтобы найти среднее значение и т.д, и по итогу получаем примерно то, что и следовало ожидать:

Method	Mean	Error	StdDev
StandartSort	3.970 us	0.0416 us	0.0369 us
BableSort	1,281.685 us	7.1960 us	6.3791 us
StartMerge	699.516 us	6.3305 us	5.9216 us

На джаве как всегда все гораздо интересней, то что называется для людей, создаем как стало привычным, через мавен, где нужно будет дописать пару строк, чтобы подгрузить джавские бенчмарки:

```
<dependency>
  <groupId>org.openjdk.jmh</groupId>
  <artifactId>jmh-core</artifactId>
  <version>1.34</version>
</dependency>
<dependency>
  <groupId>org.openjdk.jmh</groupId>
  <artifactId>jmh-generator-annprocess</artifactId>
  <version>1.34</version>
  <scope>provided</scope>
</dependency>
```

Затем прописываем начальные условия для тестирования, и перед каждым методом, что будем тестировать объявляем пару параметров тестирования:

```

@State(Scope.Benchmark)
public class Sorts {
    private static Random rnd = new Random()
    @Param({"100", "10000"})
    private static int arrSize;
    private int[] array;

    @Setup(Level.Invocation)
    public void setUp(){
        array=rnd.ints(arrSize).toArray();
    }

    @Benchmark
    @Fork(value = 1)
    @Warmup(iterations = 1)
    @Measurement(iterations = 1)
    @BenchmarkMode(Mode.AverageTime)
    @OutputTimeUnit(TimeUnit.NANOSECONDS)
    public void StandardSort(){
        Arrays.sort(array);
    }
}

```

После чего запустив мэйн:

```

package banch;

import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
        org.openjdk.jmh.Main.main(args);
    }
}

```

сможем получить довольно ожидаемую статистику:

Benchmark	(arrSize)	Mode	Cnt	Score	Error	Units
Sorts.BubbleSort	100	avgt		14003,621		ns/op
Sorts.BubbleSort	10000	avgt		107693330,108		ns/op
Sorts.HeapSort	100	avgt		3804,212		ns/op
Sorts.HeapSort	10000	avgt		693380,972		ns/op
Sorts.StandardSort	100	avgt		1841,103		ns/op
Sorts.StandardSort	10000	avgt		357457,495		ns/op

## Задание 5

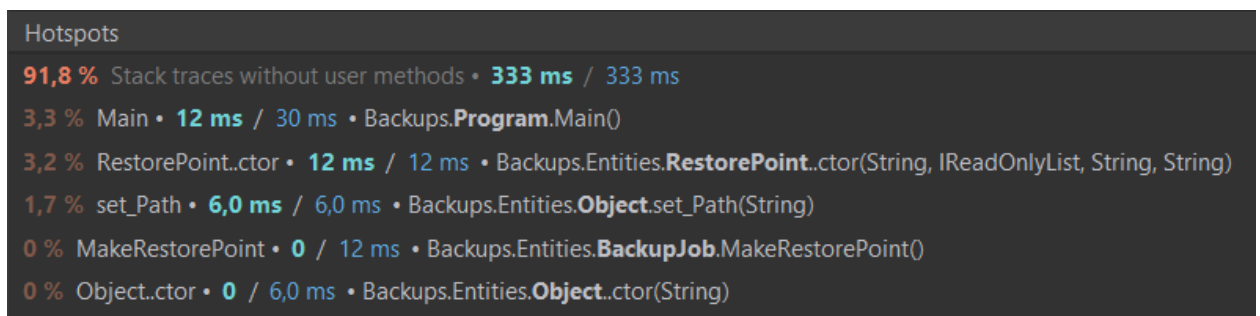
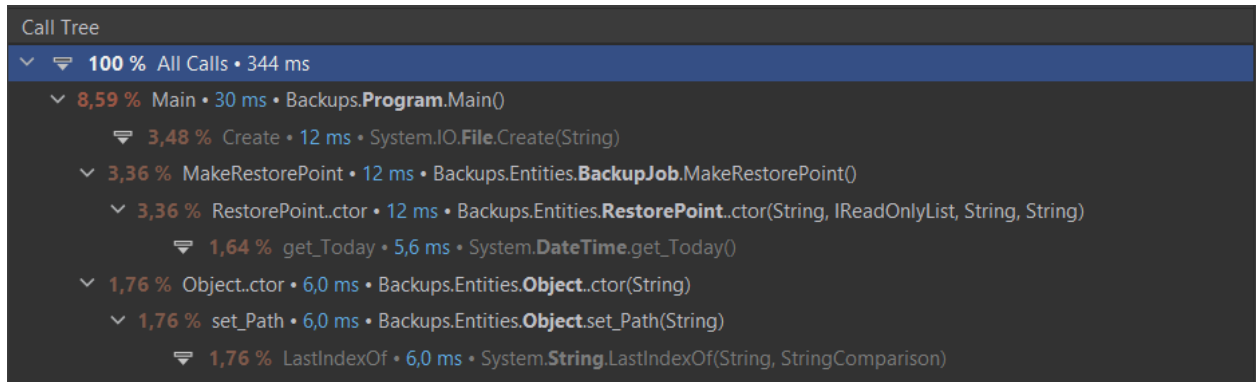
В шарпе есть прекрасный доттрэйс, который в одно нажатие позволяет нам узнать, что и на сколько долго выполнялось, в первом случае мы видим что создание стореджа, что в моем случае являлось записью архивов на комп, занимает относительно много времени

100 %	All Calls	3 367 ms
20,4 %	Main	686 ms • Backups. <b>Program</b> .Main()
19,7 %	MakeRestorePoint	662 ms • Backups.Entities. <b>BackupJob</b> .MakeRestorePoint()
19,1 %	MakeStorage	644 ms • Backups.Entities. <b>SplitStorage</b> .MakeStorage(List, String, IRepository, String)
9,04 %	Dispose	304 ms • System.IO.Compression. <b>ZipArchive</b> .Dispose()
7,42 %	WriteFile	250 ms • Backups.Entities. <b>LocalRepository</b> .WriteFile(String, Byte[], String)
4,76 %	InternalWriteAllBytes	160 ms • System.IO. <b>File</b> .InternalWriteAllBytes(String, Byte[])
1,95 %	CreateDirectory	66 ms • System.IO. <b>Directory</b> .CreateDirectory(String)
0,71 %	WriteAllBytes	24 ms • System.IO. <b>File</b> .WriteAllBytes(String, Byte[])
1,24 %	ReadAllBytes	42 ms • System.IO. <b>File</b> .ReadAllBytes(String)
1,24 %	CreateEntry	42 ms • System.IO.Compression. <b>ZipArchive</b> .CreateEntry(String)
0,18 %	ZipArchive..ctor	6,0 ms • System.IO.Compression. <b>ZipArchive</b> ..ctor(Stream, ZipArchiveMode)
0,18 %	ToString	6,1 ms • System. <b>Int32</b> .ToString()
0,18 %	RestorePoint..ctor	5,9 ms • Backups.Entities. <b>RestorePoint</b> ..ctor(String, IReadOnlyList, String, String)
0,18 %	get_Today	5,9 ms • System. <b>DateTime</b> .get_Today()
0,36 %	Create	12 ms • System.IO. <b>File</b> .Create(String)
0,18 %	RemoveObject	6,0 ms • Backups.Entities. <b>JobObjects</b> .RemoveObject(Object)
0,18 %	Remove	6,0 ms • System.Collections.Generic. <b>List`1</b> .Remove(T)
0,18 %	Object..ctor	6,0 ms • Backups.Entities. <b>Object</b> ..ctor(String)
0,18 %	set_Path	6,0 ms • Backups.Entities. <b>Object</b> .set_Path(String)
0,18 %	LastIndexOf	6,0 ms • System. <b>String</b> .LastIndexOf(String, StringComparison)

Hotspots

79,9 %	Stack traces without user methods	• 2 720 ms / 2 720 ms
11,6 %	MakeStorage	• 394 ms / 644 ms • Backups.Entities.SplitStorage.MakeStorage(List, String, IRepository, String)
7,3 %	WriteFile	• 250 ms / 250 ms • Backups.Entities.LocalRepository.WriteFile(String, Byte[], String)
0,4 %	MakeRestorePoint	• 12 ms / 662 ms • Backups.Entities.BackupJob.MakeRestorePoint()
0,4 %	Main	• 12 ms / 686 ms • Backups.Program.Main()
0,2 %	RemoveObject	• 6,0 ms / 6,0 ms • Backups.Entities.JobObjects.RemoveObject(Object)
0,2 %	set_Path	• 6,0 ms / 6,0 ms • Backups.Entities.Object.set_Path(String)
0,2 %	RestorePoint..ctor	• 5,9 ms / 5,9 ms • Backups.Entities.RestorePoint..ctor(String, IReadOnlyList, String, String)
0 %	Object..ctor	• 0 / 6,0 ms • Backups.Entities.Object..ctor(String)

Здесь же мы не записываем сторадж и сразу же все происходит на порядок быстрее



## Общий вывод:

Если вам когда-нибудь по какой-нибудь причине покажется уместным взять и использовать код из одного языка в другом, эти мысли нужно гнать из головы, как чумных из города. И если вдруг есть какая-нибудь библиотека, но она написано на другом языке, не поленитесь, перепишите, попробуйте её запустить себе дороже.