

# ПРИМЕНЕНИЕ МЕТОДА ЧЕНГСИ-ВАНГА ДЛЯ ОБФУСКАЦИИ ФУНКЦИОНАЛЬНЫХ ЯЗЫКОВ

СТУДЕНТ: МАРТЪЯНОВ В.Д.

ГРУППА: ИТ-41 БО

НАУЧНЫЙ РУКОВОДИТЕЛЬ: БАШКИН В.А.

# ВВЕДЕНИЕ

- Защита информации - является острой проблемой в наши дни.
- Обфускация - один из способов борьбы с ней в IT-сфере.

# ПОСТАНОВКА ЗАДАЧИ

- Изучить понятие обфускации
- Изучить открытую проблему обфускации функциональных программ
- Изучить и модернизировать Алгоритм Ченгси-Ванга
- Создать программу-обфускатор, которая будет защищать код функционального языка программирования на примере языка SML

# ОБФУСКАЦИЯ ПРОГРАММ

- Обфускация - запутывание кода, затрудняющее анализ и понимание алгоритмов работы программы.
- Обфускатор – программа, выполняющая обфускацию.
- Три уровня обфускации :
  1. На уровне алгоритма
  2. Запутывание исходного кода
  3. Запутывание ассемблерного кода

# ЦЕЛИ ОБФУСКАЦИИ

- Затруднение декомпиляции и изучения программ с целью обнаружения функциональности.
- Затруднение декомпиляции проприетарных программ с целью предотвращения обратной разработки или обхода DRM и систем проверки лицензий.
- Оптимизация программы с целью уменьшения размера работающего кода.

# ОТКРЫТАЯ ПРОБЛЕМА С ОБФУСКАЦИЕЙ ФУНКЦИОНАЛЬНЫХ ПРОГРАММ

- Нет аналогов

# ИСПОЛЬЗОВАНИЕ ФУНКЦИОНАЛЬНЫХ ЯЗЫКОВ

- Erlang - [Facebook](#) - бэкенд для чата
- Erlang - Серверное программное обеспечение [WhatsApp](#)
- R – Big data, Data Science
- Другие языки : F# (Microsoft), Haskell, SML

# АЛГОРИТМ ЧЕНГСИ-ВАНГА ДЛЯ ИМПЕРАТИВНЫХ ПРОГРАММ

- Создание графа потока управления этой процедуры
- Нумерация всех блоков в графе, и добавление в код процедуры переменной хранящей номер следующего выполняемого блока
- Приведение графа к однородному виду



# АЛГОРИТМ ЧЕНГСИ-ВАНГА ДЛЯ ИМПЕРАТИВНЫХ ПРОГРАММ: НАГЛЯДНО

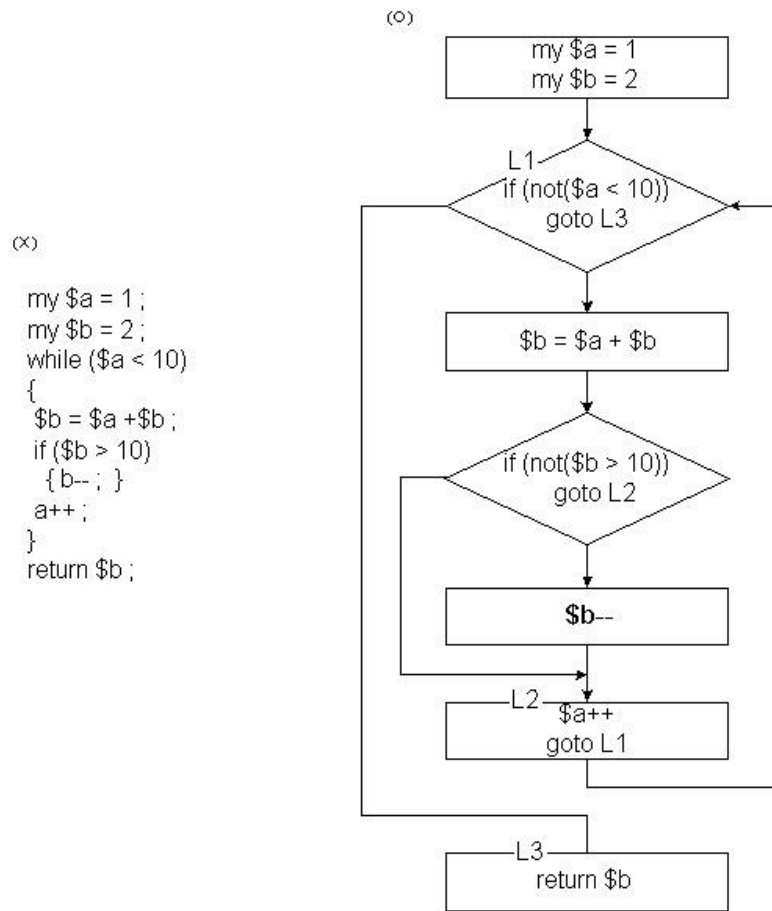


Рис.1

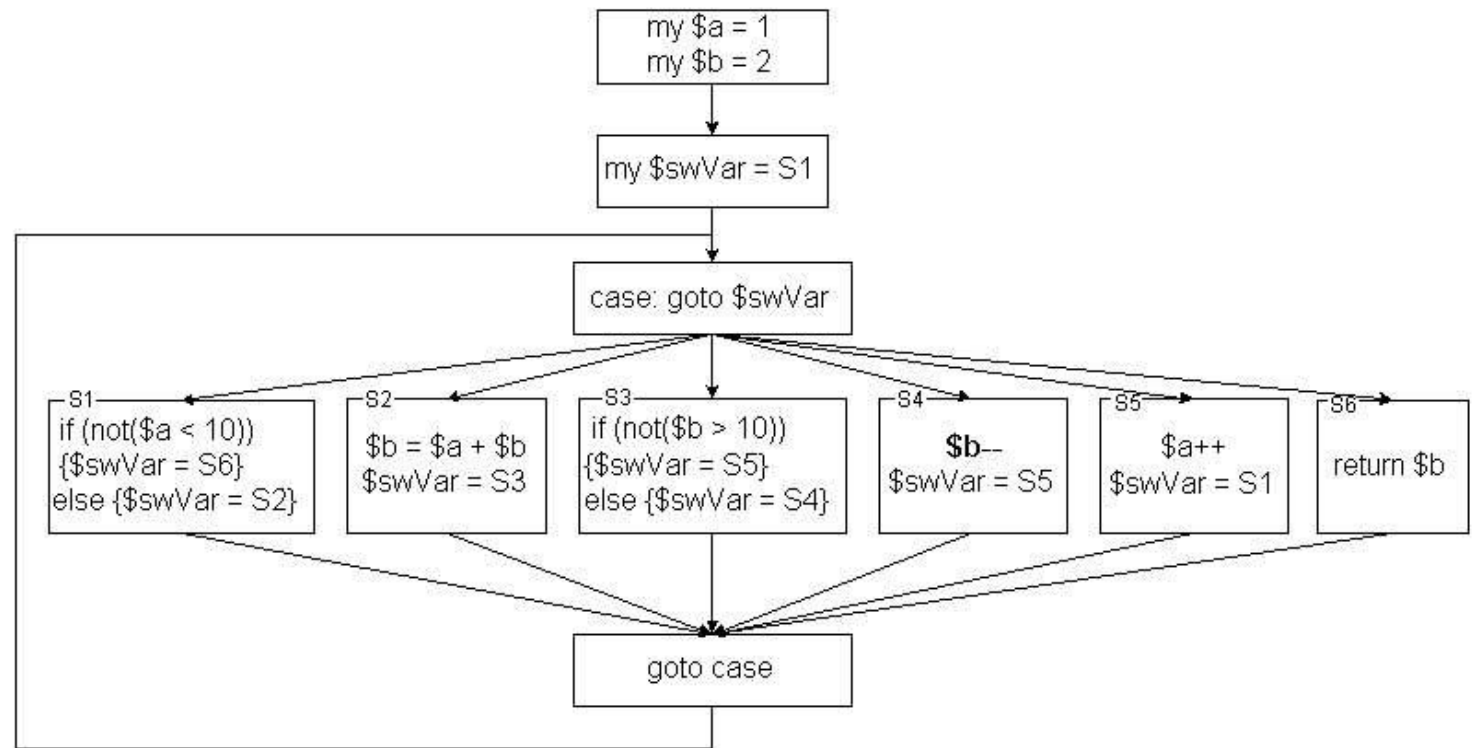


Рис.2

# АЛГОРИТМ ЧЕНГСИ-ВАНГА ДЛЯ ИМПЕРАТИВНЫХ ПРОГРАММ: ПРИМЕР РАБОТЫ

## Входные данные:

```
public class ExampleClass extends ArrayList {  
    private static Integer instance = null;  
  
    public static void main(String[] args) { hello();  
    world(); }  
  
    private static void hello() { System.out.print("Hello  
"); }  
  
    private static void world(){  
    System.out.print("World!!!"); }  
  
}
```

## Обфусцированный код:

```
public class ExampleClass extends  
ArrayList{private static Integer instance =  
null;public static void main(String[]  
args){java.util.Stack stack = new  
java.util.Stack<>(); stack.push(1);int  
postNumber;  
while(stack.size()>0){postNumber =  
stack.pop(); switch(postNumber){case  
1:stack.push(4);stack.push(2);break;case 4:  
stack.push(5); stack.push(3); break;case 5:  
break;case  
2:System.out.print("Hello");break;case 3:  
System.out.print("World!!!"); break;}}}}
```

# МОДЕРНИЗАЦИЯ АЛГОРИТМА ЧЕНГСИ-ВАНГА : ОБЩАЯ ИДЕЯ

- Введение дополнительного аргумента – аналога «переменной состояния»
- Сведение всех функций и ветвей функций в одну единую функцию, где вариант исполнения определяется значением нового аргумента
- Добавление недостижимых ветвей

# ПРИМЕР РАБОТЫ МОДЕРНИЗИРОВАННОГО АЛГОРИТМА

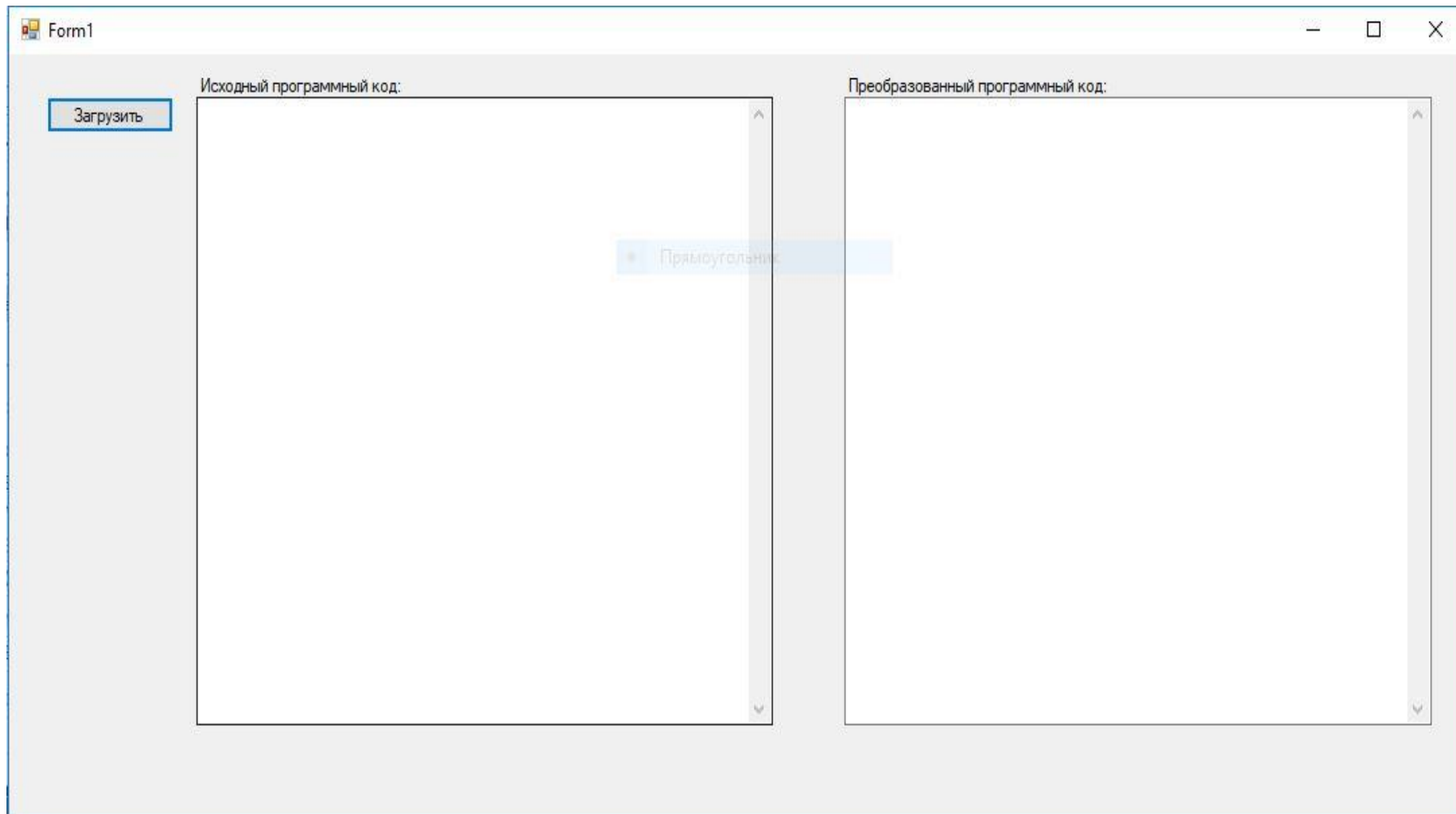
```
fun sort nil = nil
|   sort [x] = [x]
|   sort (h::t) =
      if h < hd(sort t)
      then (h::(sort t))
      else (hd(sort t))::(sort (h::(tl(sort t))));
```

1. Введение переменной состояния.
2. Добавление недостижимых ветви.



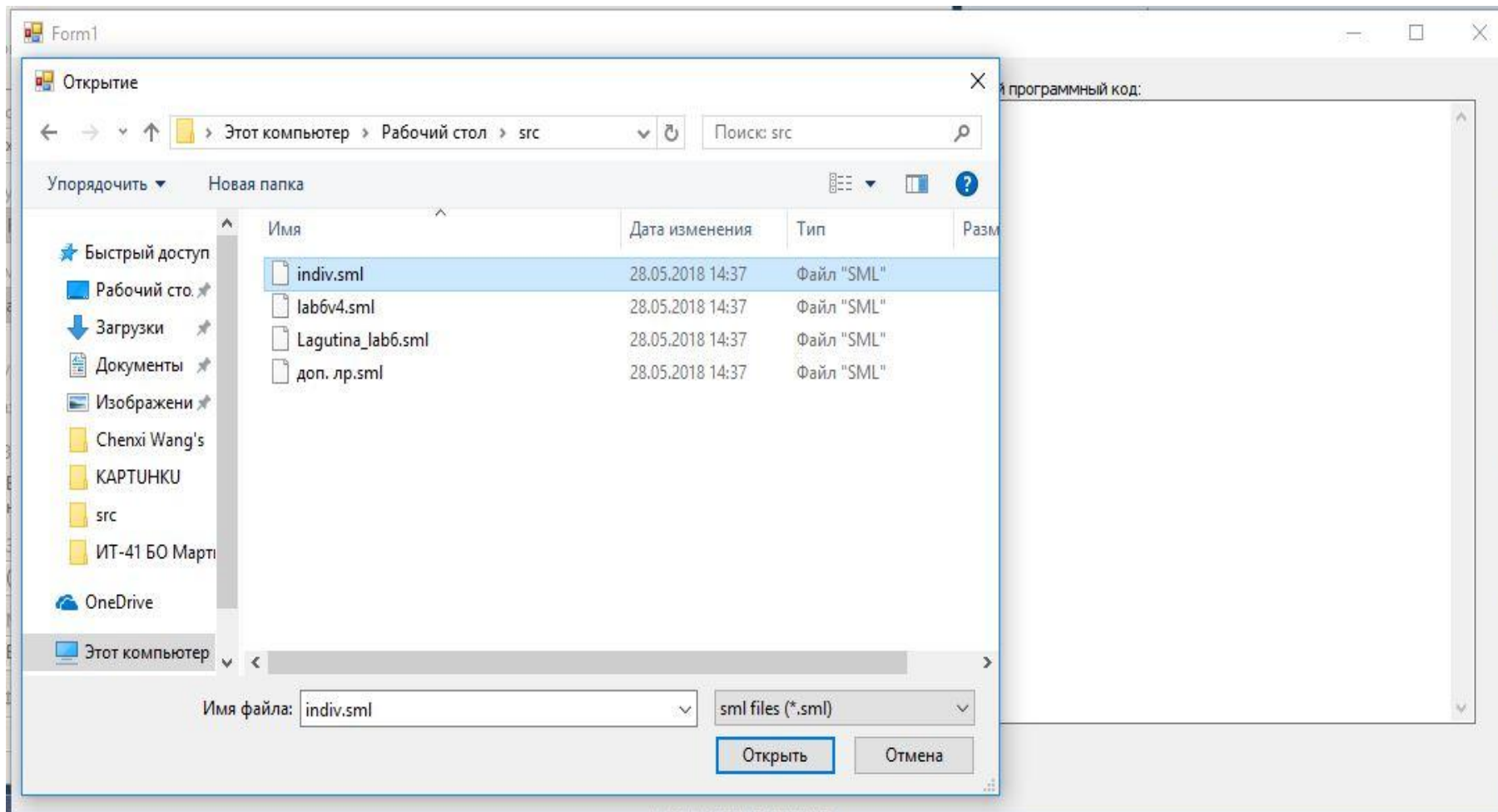
```
fun s 1 nil = nil
|   s 1 [x] = [x]
|   s 1 (h::t) = s 2 (h::t)
|   s 2 nil = [1, 2, 3]      /* недостижимая
ветка */
|   s 2 [x] = s 5 [x, x]    /* недостижимая
ветка */
|   s 2 (h::t) =
      if h < (hd(s 1 t))
      then (h::(s 1 t))
      else (hd(s 1 t))::(s 1 (h::tl(s 1 t)))
|   s x y = nil;          /* недостижимая ветка */
```

# ПРИМЕР РАБОТЫ ПРОГРАММЫ



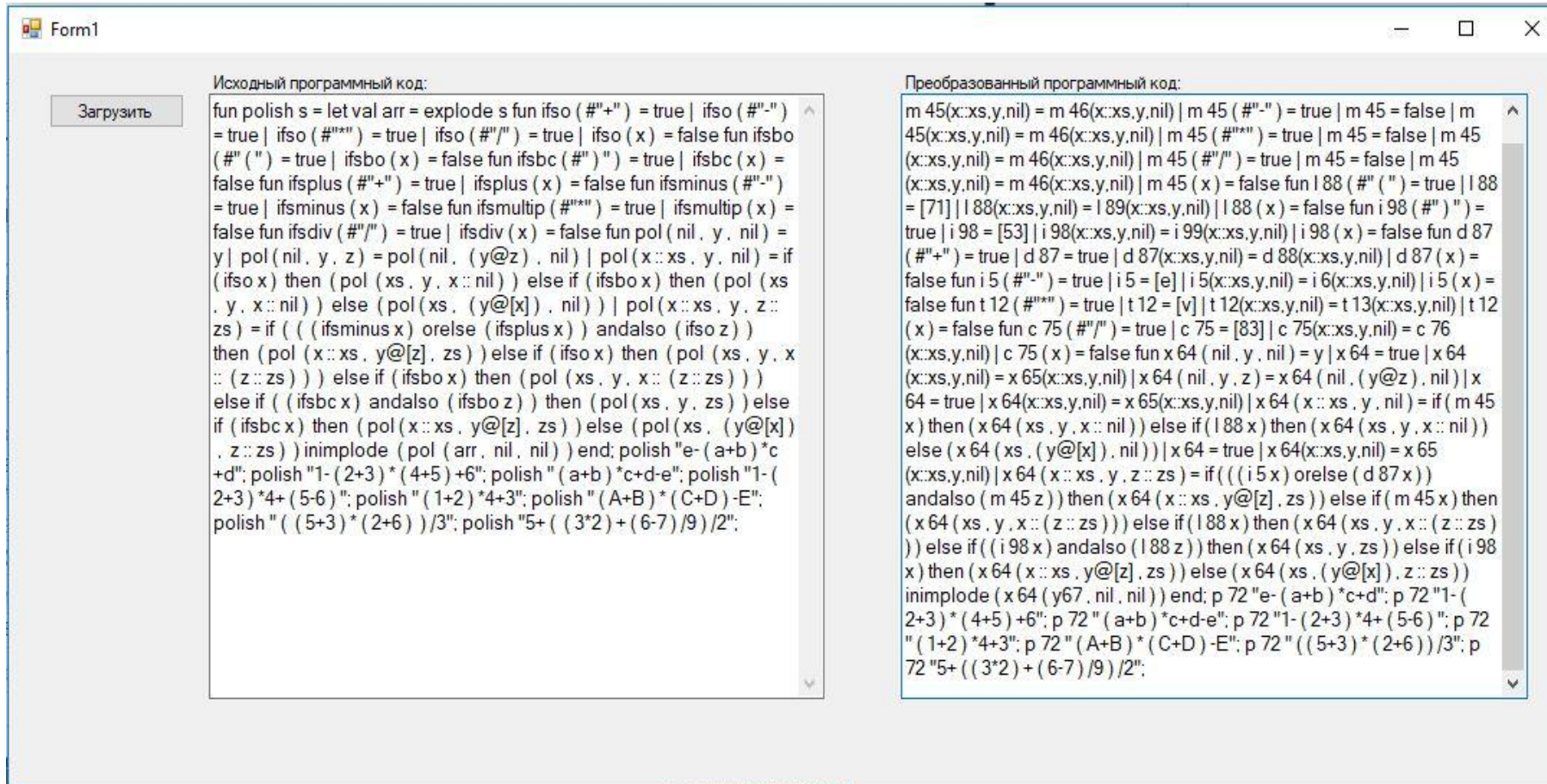
**Рис.3.**  
Интерфейс.

# ПРИМЕР РАБОТЫ ПРОГРАММЫ



**Рис.4.**  
Загрузка файла.

# ПРИМЕР РАБОТЫ ПРОГРАММЫ



Form1

Загрузить

Исходный программный код:

```
fun polish s = let val arr = explode s fun ifso (#"+") = true | ifso (#"-") = true | ifso (#"***") = true | ifso (#"/") = true | ifso (x) = false fun ifsbo (#"(") = true | ifsbo (x) = false fun ifsbc (#"") = true | ifsbc (x) = false fun ifsplus (#"+") = true | ifsplus (x) = false fun ifsminus (#"-") = true | ifsminus (x) = false fun ifsmultip (#"***") = true | ifsmultip (x) = false fun ifsddiv (#"/") = true | ifsddiv (x) = false fun pol (nil, y, nil) = y | pol (nil, y, z) = pol (nil, (y@[z]), nil) | pol (x::xs, y, nil) = if (ifso x) then (pol (xs, y, x::nil)) else if (ifsbo x) then (pol (xs, y, x::nil)) else (pol (xs, (y@[x]), nil)) | pol (x::xs, y, z::zs) = if ((ifsminus x) orelse (ifsplus x)) andalso (ifso z) then (pol (x::xs, y@[z], zs)) else if (ifso x) then (pol (xs, y, x::zs)) else if ((ifsbc x) andalso (ifsbo z)) then (pol (xs, y, zs)) else if (ifsbc x) then (pol (x::xs, y@[z], zs)) else (pol (xs, (y@[x]), z::zs)) inimplode (pol (arr, nil, nil)) end; polish "e-(a+b)*c+d"; polish "1-(2+3)*(4+5)+6"; polish "(a+b)*c+d-e"; polish "1-(2+3)*4+(5-6)"; polish "(1+2)*4+3"; polish "(A+B)*(C+D)-E"; polish "((5+3)*(2+6))/3"; polish "5+((3*2)+(6-7)/9)/2";
```

Преобразованный программный код:

```
m 45(x::xs,y,nil) = m 46(x::xs,y,nil) | m 45 (#"-") = true | m 45 = false | m 45(x::xs,y,nil) = m 46(x::xs,y,nil) | m 45 (#"***") = true | m 45 = false | m 45(x::xs,y,nil) = m 46(x::xs,y,nil) | m 45 (#"/") = true | m 45 = false | m 45(x::xs,y,nil) = m 46(x::xs,y,nil) | m 45 (x) = false fun l 88 (#"(") = true | l 88 = [71] | l 88(x::xs,y,nil) = l 89(x::xs,y,nil) | l 88 (x) = false fun i 98 (#"") = true | i 98 = [53] | i 98(x::xs,y,nil) = i 99(x::xs,y,nil) | i 98 (x) = false fun d 87 (#"+") = true | d 87 = true | d 87(x::xs,y,nil) = d 88(x::xs,y,nil) | d 87 (x) = false fun i 5 (#"-") = true | i 5 = [e] | i 5(x::xs,y,nil) = i 6(x::xs,y,nil) | i 5 (x) = false fun t 12 (#"***") = true | t 12 = [v] | t 12(x::xs,y,nil) = t 13(x::xs,y,nil) | t 12 (x) = false fun c 75 (#"/") = true | c 75 = [83] | c 75(x::xs,y,nil) = c 76 (x::xs,y,nil) | c 75 (x) = false fun x 64 (nil, y, nil) = y | x 64 = true | x 64 (x::xs,y,nil) = x 65(x::xs,y,nil) | x 64 (nil, y, z) = x 64 (nil, (y@[z]), nil) | x 64 = true | x 64(x::xs,y,nil) = x 65(x::xs,y,nil) | x 64 (x::xs, y, nil) = if (m 45 x) then (x 64 (xs, y, x::nil)) else if (l 88 x) then (x 64 (xs, y, x::nil)) else (x 64 (xs, (y@[x]), nil)) | x 64 = true | x 64(x::xs,y,nil) = x 65 (x::xs,y,nil) | x 64 (x::xs, y, z::zs) = if (((i 5 x) orelse (d 87 x)) andalso (m 45 z)) then (x 64 (x::xs, y@[z], zs)) else if (m 45 x) then (x 64 (xs, y, x::zs)) else if (l 88 x) then (x 64 (xs, y, x::zs)) else if ((i 98 x) andalso (l 88 z)) then (x 64 (xs, y, zs)) else if (i 98 x) then (x 64 (x::xs, y@[z], zs)) else (x 64 (xs, (y@[x]), z::zs)) inimplode (x 64 (y67, nil, nil)) end; p 72 "e-(a+b)*c+d"; p 72 "1-(2+3)*(4+5)+6"; p 72 "(a+b)*c+d-e"; p 72 "1-(2+3)*4+(5-6)"; p 72 "(1+2)*4+3"; p 72 "(A+B)*(C+D)-E"; p 72 "((5+3)*(2+6))/3"; p 72 "5+((3*2)+(6-7)/9)/2";
```

Рис.5.  
Вывод  
результата.

# ЗАКЛЮЧЕНИЕ

В ходе работы была рассмотрена такая актуальная в наш век тема, как защита информации. Было изучено понятие «обфускации». Рассмотрены виды этого понятия. Узнали открытую проблему и применение функциональных языков. Был модернизирован один из наиболее известных алгоритмов «защиты кода». В результате работы получилась простая программа-обфускатор, с помощью которой можно запутать код функционального языка программирования SML. Что означает возможность решения такой нелегкой проблемы, с которой сталкиваются ежедневно огромное количество работников IT-сферы.



# СПИСОК ЛИТЕРАТУРЫ

- **Башкин В.А. Функциональное программирование на языке SML// метод. указания/** В.А. Башкин; Яросл. гос. ун-т. – Ярославль, ЯрГУ, 2007 г.
- **Чернов, Л. В. Анализ запутывающих преобразований программ//Л.В. Чернов//Труды Института системного программирования РАН. Том 3, 2002 г. стр. 7-38.**
- **Rollcs, R. Unpacking virtualization obfuscators/R. Rollcs.//In Proc. 3rd USENIX Workshop on Offensive Technologies (WOOT'09), August 2009.**
- **А.Ю.Тихонов, Л.И.Аветисян, В.А.Иадарян Методика извлечения алгоритма из бинарного кода на основе динамического анализа// Проблемы информационной безопасности. Компьютерные системы. — 2008. — Т. №3. — С. 66-71.**



СПАСИБО ЗА ВНИМАНИЕ