МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
Кафедра информационных систем

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1
по курсу: Теория информационных процессов и систем
Парадигмы и языки программирования

Работу выполнили студенты:
Трофимов Влад
группа 3511
Виноградов Павел
группа 3511

САНКТ-ПЕТЕРБУРГ
2015

В ходе лабораторной работы был реализован алгоритм бинарного поиска на 4 языках программирования и нарисованы графы управления.

Исходный код:

Язык C:

```
#include <stdio.h>
#include <stdlib.h>

int cmpint(int *pa, int pb)
{
	int a = *pa;
	int b = pb;
	if (a < b) return -1;
	if (a == b) return 0;
	if (a > b) return 1;
}

int bsearch(int Array[], int n, int key,int(*cmp)(int *, int pb))
{
	unsigned left = 1, right = n;
	int NotFound = -1;
	if (!(Array && n > 0 && key && cmp))
		return NotFound;

	while (left < right)
	{
		unsigned m = (left + right) / 2;

		if (cmp(Array + m, key) < 0)
			left = m + 1;
		else
			right = m;
	}
	return (cmp(Array + right, key) == 0) ? right : NotFound;
}

int main(void)
{
	int a[] = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024};
	int n = sizeof(a) / sizeof(*a) - 1;

	int key = 4;

	int b = bsearch(a, n, key,cmpint);
	printf("Position of %d in [ ", key);
	for (int i = 0; i <= n; i++)
	{
		printf("%d ", a[i]);
	}
	printf("] is %d\n", b);

	key = 0;

	b = bsearch(a, n, key, cmpint);
	printf("Position of %d in [ ", key);
	for (int i = 0; i <= n; i++)
	{
		printf("%d ", a[i]);
	}
	printf("] is %d\n", b);
	return 0;
}
```

Язык Java:

```
import java.util.Arrays;
import java.util.List;

public class Bsearch {
	public static void main(String[] args) {
		List<Integer> list = Arrays.asList(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024);
		Integer[] array = (Integer[])list.toArray();
		int value = 4;
		int result = new ExtendedArray(array).binarySearch(value);
		System.out.println("Position of " + value + " in " + list + " is " + result);

		value = 0;
		result = new ExtendedArray(array).binarySearch(value);
		System.out.println("Position of " + value + " in " + list + " is " + result);
	}
}

class ExtendedArray {
	private Integer[] data;

	public ExtendedArray(Integer[] source) {
		this.data = source;
	}

	public int binarySearch(int value){
	 int h = data.length - 1;
	 int l = 0;
	 while (h >= l) {
			int m = l + ((h - l) / 2);
			if (data[m] > value) {
				h = m - 1;
			} else if (data[m] < value) {
				l = m + 1;
			} else {
				return m;
			}
	 }
	 return -1;
	}
}
```

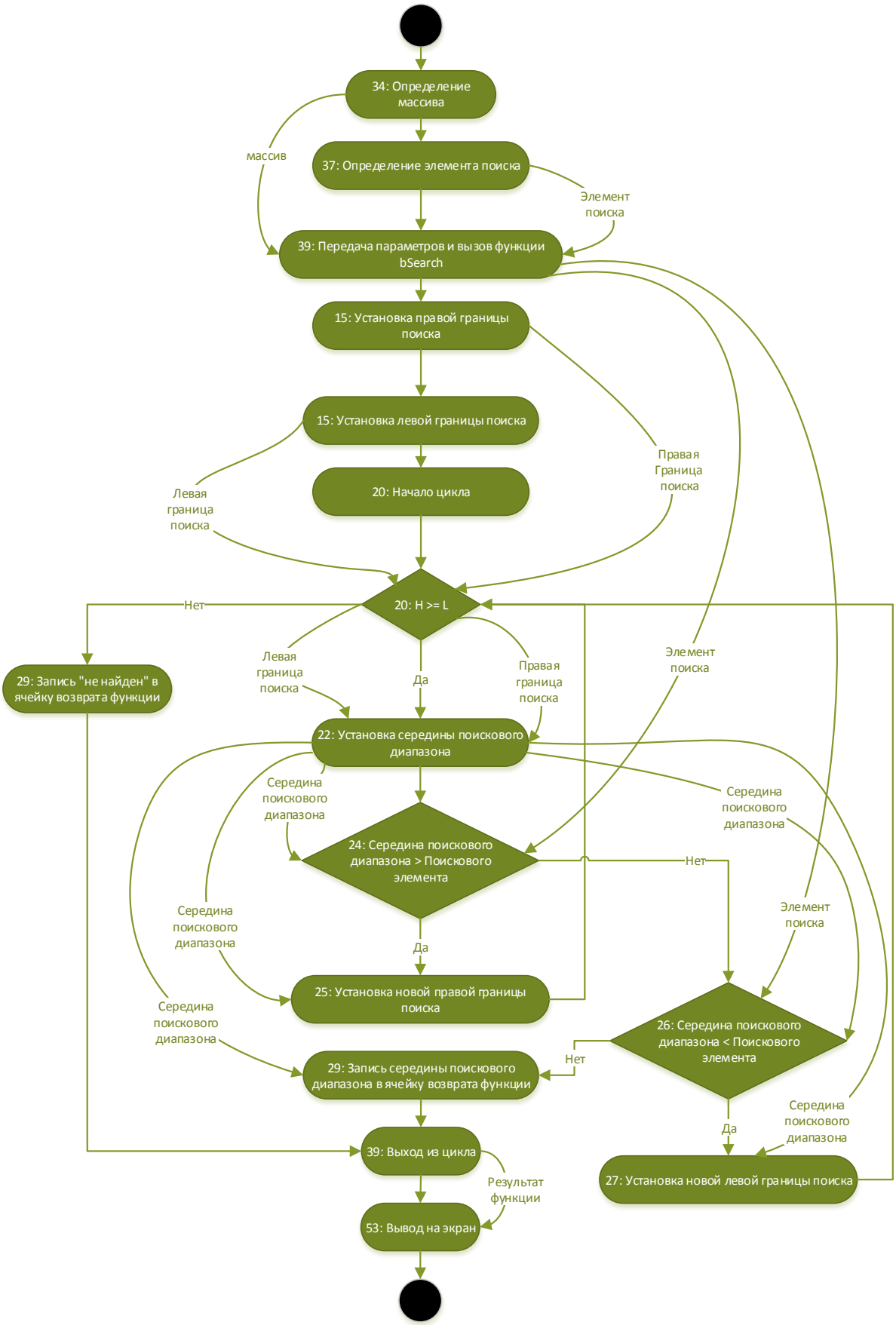
Язык Common Lisp:

```
(defun binary-search (value array &optional (low 0) (high (1- (length array))))
  (if (< high low)
      nil
      (let ((middle (floor (/ (+ low high) 2))))

        (cond ((> (aref array middle) value)
               (binary-search value array low (1- middle)))

              ((< (aref array middle) value)
               (binary-search value array (1+ middle) high))

              (t middle)))))

(setq array (make-array '(11)
                        :initial-contents
                        '(1 2 4 8 16 32 64 128 256 512 1024)))
(setq value 4)
(setq result (binary-search value array))
(format t "Position of ~S~%in ~S~%is ~S~%" value array result)

(setq value 0)
(setq result (binary-search value array))
(format t "Position of ~S~%in ~S~%is ~S~%" value array result)
```

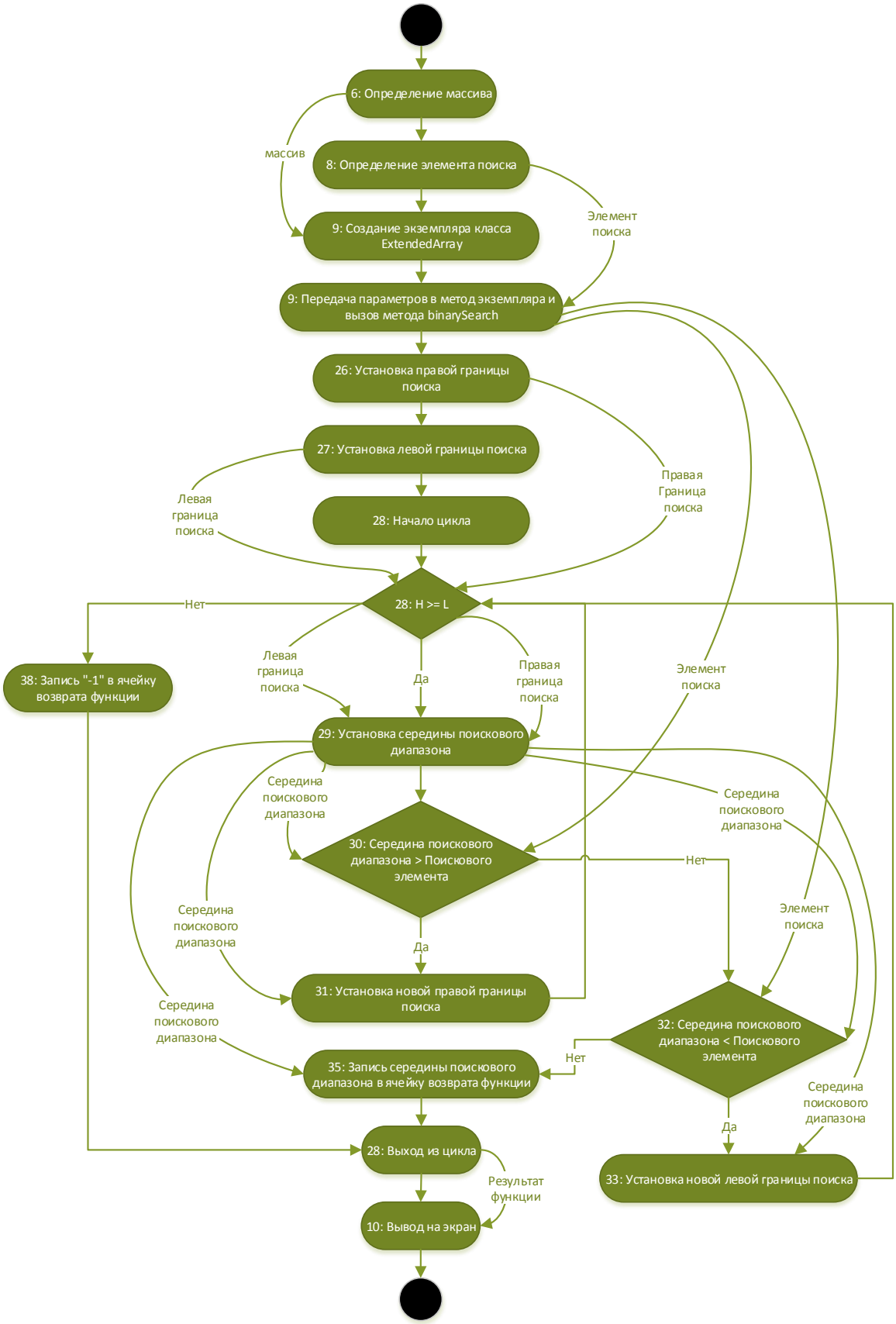
Язык Prolog:

```
bin_search(Elt,List,Result):-
  length(List,N), bin_search_inner(Elt,List,1,N,Result).

bin_search_inner(Elt,List,J,J,J):-
  nth(J,List,Elt).
bin_search_inner(Elt,List,Begin,End,Mid):-
  Begin < End,
  Mid is (Begin+End) div 2,
  nth(Mid,List,Elt).
bin_search_inner(Elt,List,Begin,End,Result):-
  Begin < End,
  Mid is (Begin+End) div 2,
  nth(Mid,List,MidElt),
  MidElt < Elt,
  NewBegin is Mid+1,
  bin_search_inner(Elt,List,NewBegin,End,Result).
bin_search_inner(Elt,List,Begin,End,Result):-
  Begin < End,
  Mid is (Begin+End) div 2,
  nth(Mid,List,MidElt),
  MidElt > Elt,
  NewEnd is Mid-1,
  bin_search_inner(Elt,List,Begin,NewEnd,Result).
```

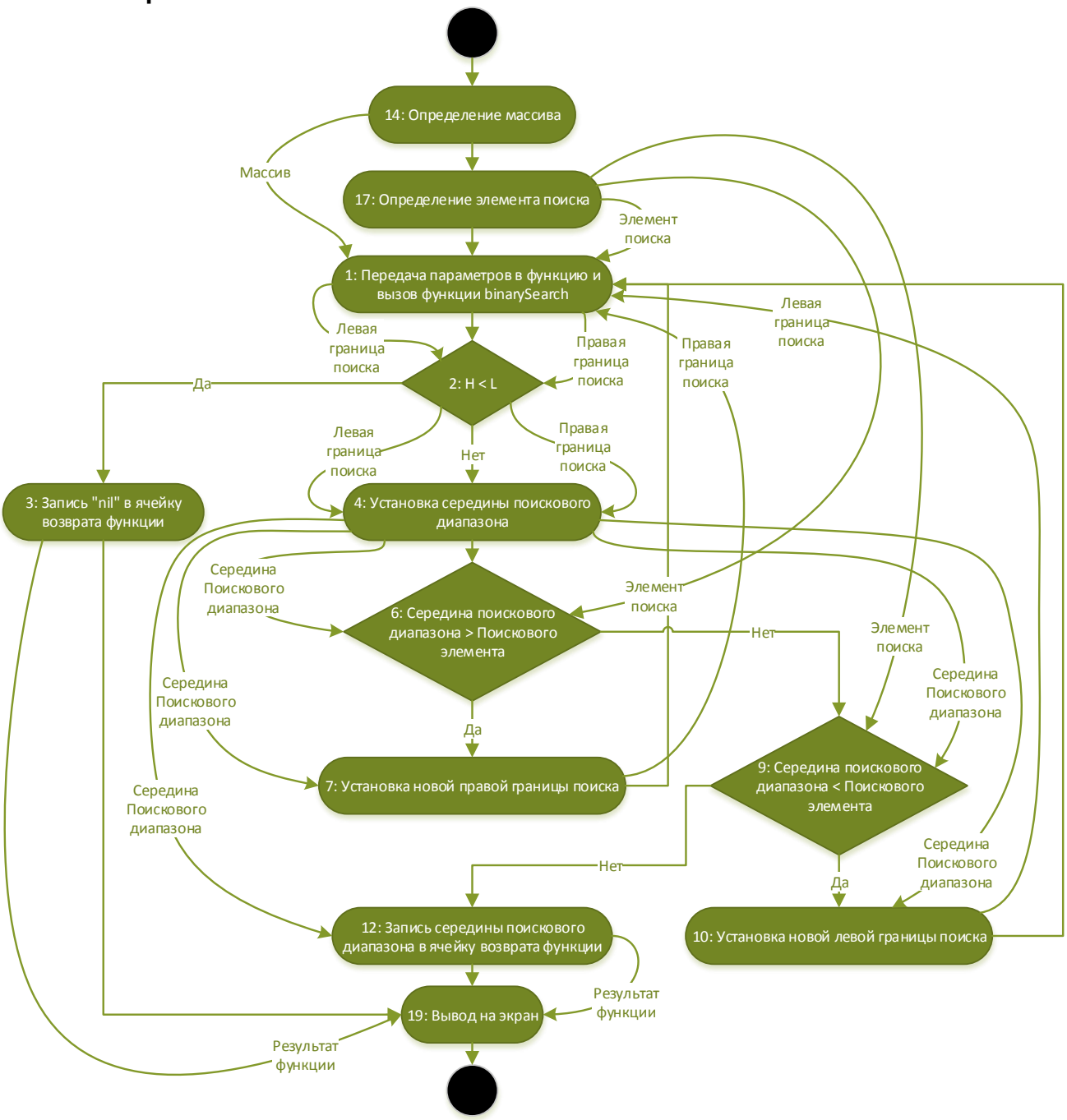
Графы:
C


34: Определение массива
37: Определение элемента поиска
39: Передача параметров и вызов функции bSearch
15: Установка правой границы поиска
15: Установка левой границы поиска
20: Начало цикла
20: H >= L
29: Запись "не найден" в ячейку возврата функции
22: Установка середины поискового диапазона
24: Середина поискового диапазона > Поискового элемента
25: Установка новой правой границы поиска
26: Середина поискового диапазона < Поискового элемента
27: Установка новой левой границы поиска
29: Запись середины поискового диапазона в ячейку возврата функции
39: Выход из цикла
53: Вывод на экран
массив
Элемент поиска
Левая граница поиска
Правая Граница поиска
Нет
Да
Середина поискового диапазона
Результат функции

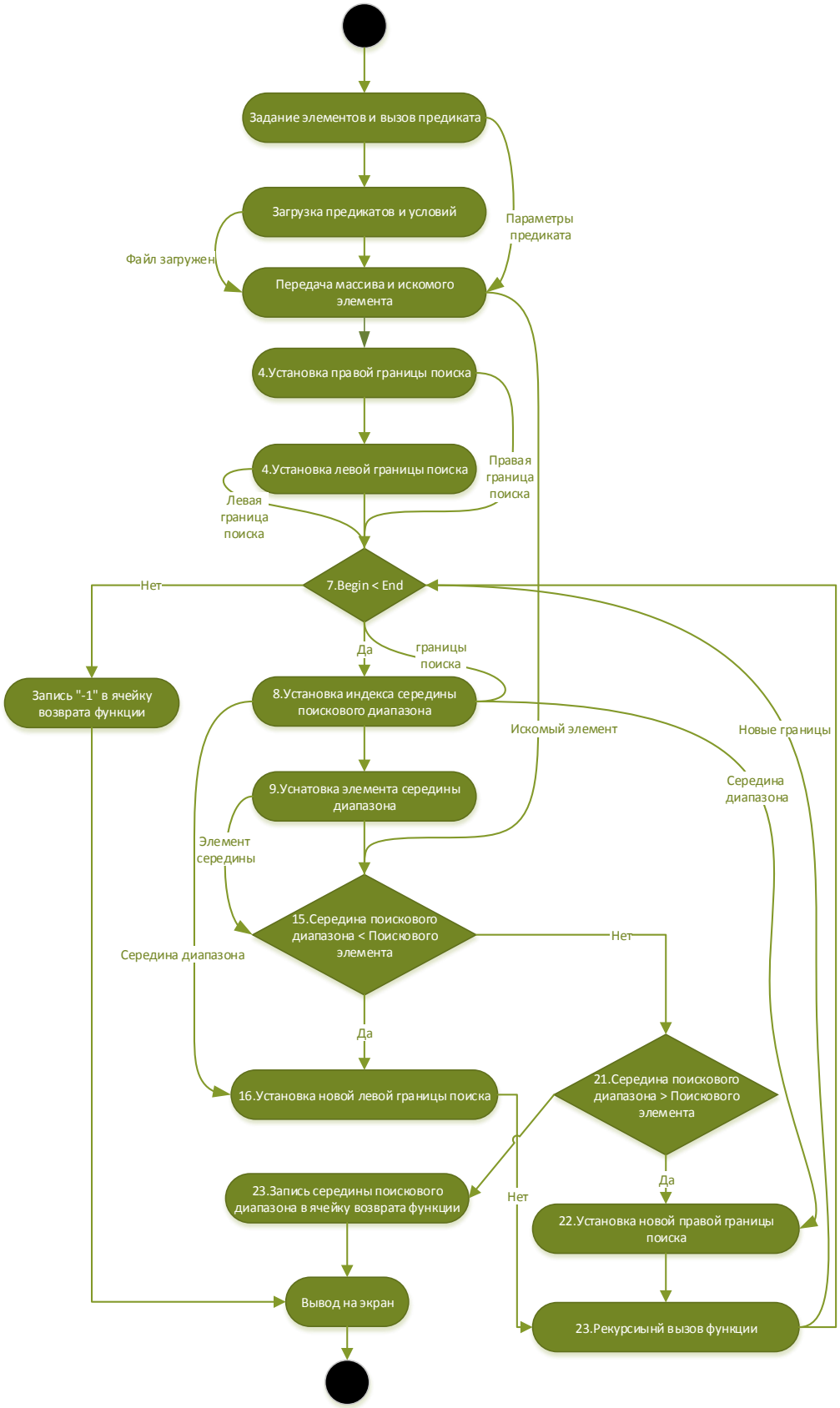
Java


6: Определение массива
массив
8: Определение элемента поиска
Элемент поиска
9: Создание экземпляра класса ExtendedArray
9: Передача параметров в метод экземпляра и вызов метода binarySearch
26: Установка правой границы поиска
27: Установка левой границы поиска
Левая граница поиска
Правая Граница поиска
28: Начало цикла
28: H >= L
Нет
38: Запись "-1" в ячейку возврата функции
Левая граница поиска
Да
Правая граница поиска
Элемент поиска
29: Установка середины поискового диапазона
Середина поискового диапазона
Середина поискового диапазона
30: Середина поискового диапазона > Поискового элемента
Нет
Середина поискового диапазона
Элемент поиска
Да
31: Установка новой правой границы поиска
Середина поискового диапазона
32: Середина поискового диапазона < Поискового элемента
Нет
35: Запись середины поискового диапазона в ячейку возврата функции
Середина поискового диапазона
Да
28: Выход из цикла
33: Установка новой левой границы поиска
Результат функции
10: Вывод на экран

Common Lisp


14: Определение массива
Массив
17: Определение элемента поиска
Элемент поиска
1: Передача параметров в функцию и вызов функции binarySearch
Левая граница поиска
Правая граница поиска
Левая граница поиска
Правая граница поиска
2: H < L
Да
Нет
Левая граница поиска
Правая граница поиска
3: Запись "nil" в ячейку возврата функции
4: Установка середины поискового диапазона
Середина Поискового диапазона
Элемент поиска
6: Середина поискового диапазона > Поискового элемента
Нет
Элемент поиска
Середина Поискового диапазона
Середина Поискового диапазона
Да
7: Установка новой правой границы поиска
9: Середина поискового диапазона < Поискового элемента
Середина Поискового диапазона
Нет
Да
Середина Поискового диапазона
12: Запись середины поискового диапазона в ячейку возврата функции
10: Установка новой левой границы поиска
Результат функции
19: Вывод на экран
Результат функции

Prolog


Задание элементов и вызов предиката
Загрузка предикатов и условий
Параметры предиката
Файл загружен
Передача массива и искомого элемента
4.Установка правой границы поиска
4.Установка левой границы поиска
Правая граница поиска
Левая граница поиска
7.Begin < End
Нет
Да
границы поиска
Запись "-1" в ячейку возврата функции
8.Установка индекса середины поискового диапазона
Искомый элемент
Новые границы
Середина диапазона
9.Уснатовка элемента середины диапазона
Элемент середины
15.Середина поискового диапазона < Поискового элемента
Середина диапазона
Нет
Да
16.Установка новой левой границы поиска
21.Середина поискового диапазона > Поискового элемента
Да
Нет
23.Запись середины поискового диапазона в ячейку возврата функции
22.Установка новой правой границы поиска
Вывод на экран
23.Рекурсиный вызов функции