Основы программирования Лабораторная работа №13

Рекурсия - 2

Власенко О.Ф.

Факториал – рекурсивная реализация

```
long fuct2(int n) {
    if (n == 0) {
        return 1;
    }

    long res = fuct2(n - 1) * n;
    return res;
}

void main() {
    int n = 4;
    long f = fuct2(n);

    printf("%d! = %ld", n, f);
}
```

Рекуррентная формула

Факториал может быть задан следующей рекуррентной формулой:

$$n! = \begin{cases} 1 & n = 0, \\ n \cdot (n-1)! & n > 0. \end{cases}$$

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

РЕКУРСИВНЫЙ СПУСК

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

РЕКУРСИВНЫЙ СПУСК

Факториал – трассировка

```
long fuct2(int n) {
	if (n == 0) {
		return 1;
	}
	long res = fuct2(n - 1) * n;
	return res;
}
```

```
void main() {
	int n = 4;
	long f = fuct2(n);
	printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

РЕКУРСИВНЫЙ СПУСК

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2

РЕКУРСИВНЫЙ СПУСК

Факториал – трассировка

```
long fuct2(int n) {
	if (n == 0) {
		return 1;
	}
	long res = fuct2(n - 1) * n;
	return res;
}
```

```
void main() {
	int n = 4;
	long f = fuct2(n);
	printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2

---

fuct2(1): n = 1

res = fuct2(0)*1

РЕКУРСИВНЫЙ СПУСК

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2

---

fuct2(1): n = 1

res = fuct2(0)*1

---

fuct2(0): n = 0

РЕКУРСИВНЫЙ СПУСК

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2

---

fuct2(1): n = 1

res = fuct2(0)*1

---

fuct2(0): n = 0

РЕКУРСИВНЫЙ СПУСК **РЕКУРСИВНЫЙ ВОЗВРАТ**

Факториал – трассировка

```
long fuct2(int n) {
	if (n == 0) {
		return 1;
	}
	long res = fuct2(n - 1) * n;
	return res;
}
```

```
void main() {
	int n = 4;
	long f = fuct2(n);
	printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2

---

fuct2(1): n = 1

res = fuct2(0)*1

---

fuct2(0): n = 0

РЕКУРСИВНЫЙ СПУСК

---

return 1

РЕКУРСИВНЫЙ ВОЗВРАТ

Факториал – трассировка

```
long fuct2(int n) {
	if (n == 0) {
		return 1;
	}
	long res = fuct2(n - 1) * n;
	return res;
}
```

```
void main() {
	int n = 4;
	long f = fuct2(n);
	printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2

---

fuct2(1): n = 1

res = fuct2(0)*1 = 1 * 1 = 1 return 1

---

fuct2(0): n = 0 return 1

РЕКУРСИВНЫЙ СПУСК **РЕКУРСИВНЫЙ ВОЗВРАТ**

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3

---

fuct2(2): n = 2

res = fuct2(1)*2 = 1 * 2 = 2 return 2

---

fuct2(1): n = 1

res = fuct2(0)*1 = 1 * 1 = 1 return 1

---

fuct2(0): n = 0 return 1

РЕКУРСИВНЫЙ СПУСК **РЕКУРСИВНЫЙ ВОЗВРАТ**

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4

---

fuct2(3): n = 3

res = fuct2(2)*3 = 2 * 3 = 6 return 6

---

fuct2(2): n = 2

res = fuct2(1)*2 = 1 * 2 = 2 return 2

---

fuct2(1): n = 1

res = fuct2(0)*1 = 1 * 1 = 1 return 1

---

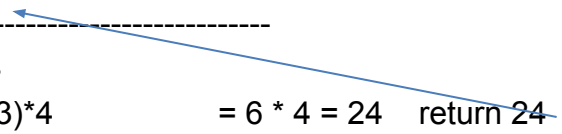
fuct2(0): n = 0 return 1

РЕКУРСИВНЫЙ СПУСК **РЕКУРСИВНЫЙ ВОЗВРАТ**

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```



main(): f = fuct2(4)

---

fuct2(4): n = 4

res = fuct2(3)*4 = 6 * 4 = 24 return 24

---

fuct2(3): n = 3

res = fuct2(2)*3 = 2 * 3 = 6 return 6

---

fuct2(2): n = 2

res = fuct2(1)*2 = 1 * 2 = 2 return 2

---

fuct2(1): n = 1

res = fuct2(0)*1 = 1 * 1 = 1 return 1

---

fuct2(0): n = 0 return 1

РЕКУРСИВНЫЙ СПУСК **РЕКУРСИВНЫЙ ВОЗВРАТ**

Факториал – трассировка

```
long fuct2(int n) {
      if (n == 0) {
            return 1;
      }
      long res = fuct2(n - 1) * n;
      return res;
}
```

```
void main() {
      int n = 4;
      long f = fuct2(n);
      printf("%d! = %ld", n, f);
}
```

main(): f = fuct2(4) = 24 `printf("%d! = %ld", 4, 24);`

---

fuct2(4): n = 4
res = fuct2(3)*4 = 6 * 4 = 24 return 24

---

fuct2(3): n = 3
res = fuct2(2)*3 = 2 * 3 = 6 return 6

---

fuct2(2): n = 2
res = fuct2(1)*2 = 1 * 2 = 2 return 2

---

fuct2(1): n = 1
res = fuct2(0)*1 = 1 * 1 = 1 return 1

---

fuct2(0): n = 0 return 1

РЕКУРСИВНЫЙ СПУСК **РЕКУРСИВНЫЙ ВОЗВРАТ**

Задача 1

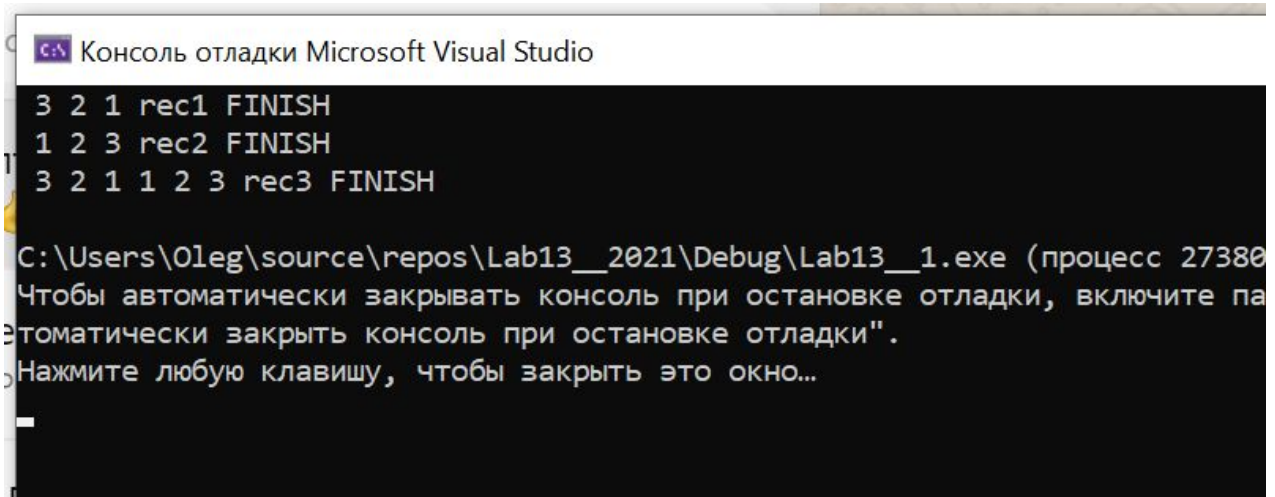
Собрать и отладить (т.е. заставить работать) код рекурсивного вычисления факториала.

И выполнить трассировку его для n=5 используя встроенный отладчик VS

Простейшие рекурсивные функции

```
void rec1(int n) {
     printf(" %d", n);
     if (n > 1) {
          rec1(n - 1);
     }
}

void rec2(int n) {
     if (n > 1) {
          rec2(n - 1);
     }
     printf(" %d", n);
}

void rec3(int n) {
     printf(" %d", n);
     if (n > 1) {
          rec3(n - 1);
     }
     printf(" %d", n);
}
```

```
void main() {
     rec1(3);
     printf(" rec1 FINISH\n");

     rec2(3);
     printf(" rec2 FINISH\n");

     rec3(3);
     printf(" rec3 FINISH\n");
}
```



Задача 2

Используя простейшие рекурсивные функции с предыдущего слайда в качестве вдохновения и основы, сделайте собственные рекурсивные функции (`f1()`, `f2()`, `f3()`), которые выводят в консоль последовательность чисел:

Задача 2.1.

Вызов функции: `f1(11)`

Формируемый вывод: `11 9 7 5 3 1`

Задача 2.2.

Вызов функции: `f2(11)`

Формируемый вывод: `1 3 5 7 9 11`

Задача 2.3.

Вызов функции: `f3(11)`

Формируемый вывод: `11 9 7 5 3 1 3 5 7 9 11`

Задача 3

Выполнить трассировку только что созданных функций

Задача 3.1.

Выполнить трассировку для n=7 используя встроенный отладчик VS. Выполнить трассировку по очереди для всех функций f1(), f2(), f3()

Задача 3.2.

Выполнить трассировку для n=7 используя встроенный бумагу и ручку/карандаш. Выполнить трассировку по очереди для всех функций f1(), f2(), f3()

Задача 4* (по мотивам ЕГЭ)

Нужно выполнить трассировку представленного кода в отладчике VS.

(Задачи, подобные этой, периодически встречаются в ЕГЭ по информатике.)

```
void recEGE1(int n) {
    if (n >= 1) {
        printf(" %d", n);
        recEGE1(n - 1);
        recEGE1(n - 1);
    }
}

void main() {
    recEGE1(3);
}
```

Задача 5* (по мотивам ЕГЭ)

Нужно выполнить трассировку представленного кода в отладчике VS.

(Крайне желательно) выполнить ручную трассировку – на бумаге!

```
//  ЕГЭ 2017 Демо -
//  Задача 11: Чему равна сумма напечатанных на экране чисел при выполнении вызова F(10) ?
F1(10);
```

```
void F1(int n) {
    if (n > 2) {
        printf("%d\n", n);
        F1(n - 3);
        F1(n - 4);
    }
}
```

Задача 6* (по мотивам ЕГЭ)

Нужно выполнить трассировку представленного кода в отладчике VS.

(Крайне желательно) выполнить ручную трассировку – на бумаге!

```
//  ЕГЭ 2015 Демо -
//  Задача 11: Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова F(1)?
F2(1);

void F2(int n)
{
    printf("%d\n", n);
    if (n < 5) {
        F2(n + 1);
        F2(n + 3);
    }
}
```

Задача 7** (по мотивам ЕГЭ)

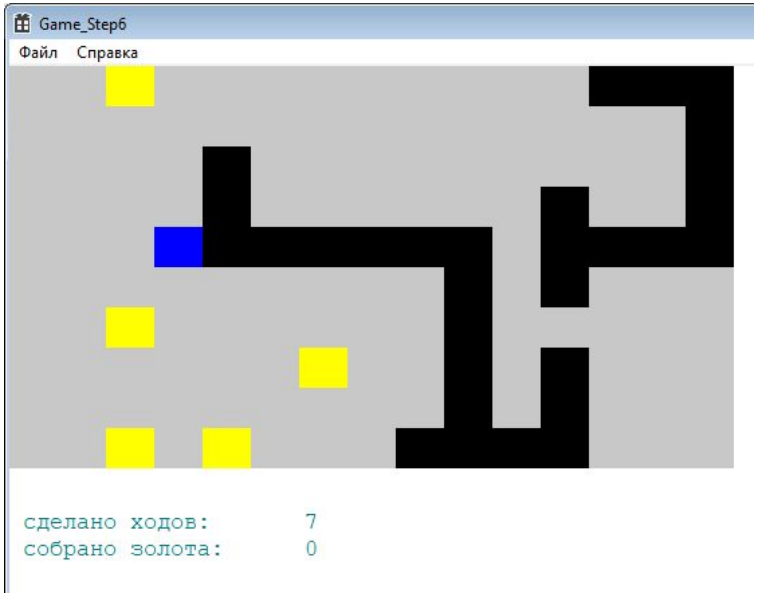
Нужно выполнить трассировку представленного кода в отладчике VS.

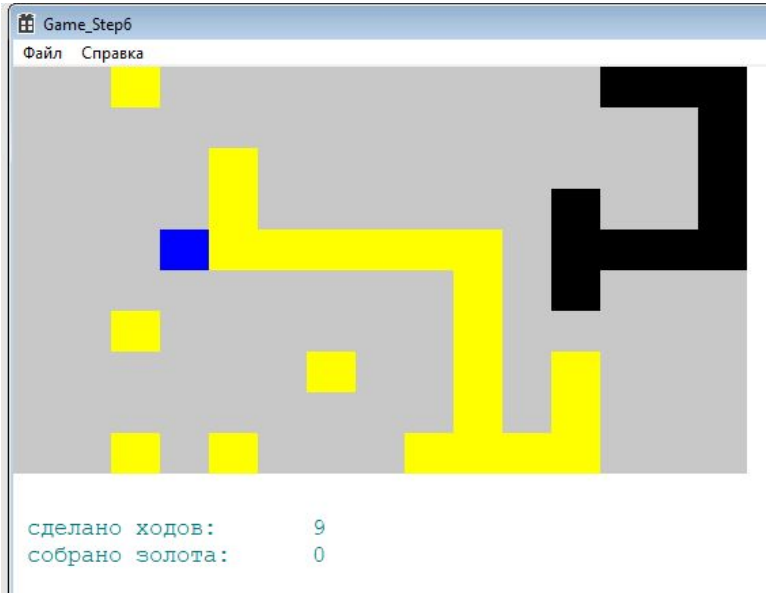
(желательно) выполнить ручную трассировку – на бумаге!

```
//  ЕГЭ 2016 Демо -
//  Задача 11: записаны две рекурсивные функции : F и G.
//  Сколько символов «звёздочка» будет напечатано на экране при выполнении вызова F(11)?
F3(11);


void G3(int n);

void F3(int n) {
    if (n > 0)
        G3(n - 1);
}

void G3(int n) {
    printf("*");
    if (n > 1)
        F3(n - 3);
}
```

Задача 8*

В игру, реализованную в лаб работах 8, 9 и т.д. добавить «руку Мидаса» - прикосновение к стене превращают всю стену в набор золотых элементов. В золото превращаются ВСЕ ЭЛЕМЕНТЫ связанные друг с другом – в не зависимости от конфигурации стены.





Задача 8* (Код обслуживающий)

```
case WM_KEYDOWN:
    switch (wParam)
    {
    case 0x4d: // M - MidasHand
        midasHandToRight();
        InvalidateRect(hWnd, NULL, TRUE);
        break;
```

```
void midasHandToRight() {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < M - 1; ++j) {
            if (a[i][j] == 3 && a[i][j + 1] == 2) {
                doMidasHand(i, j + 1);
            }
        }
    }
}
```

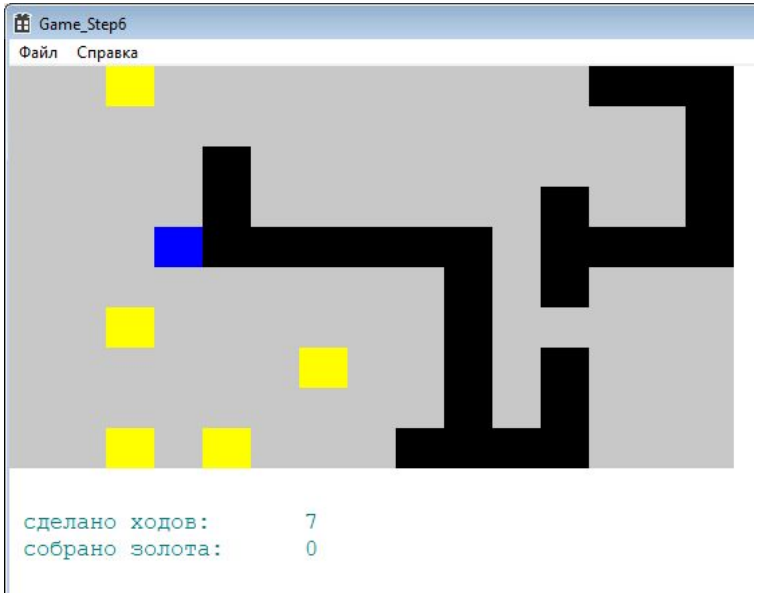
Задача 8* (Код превращения стены в золото)

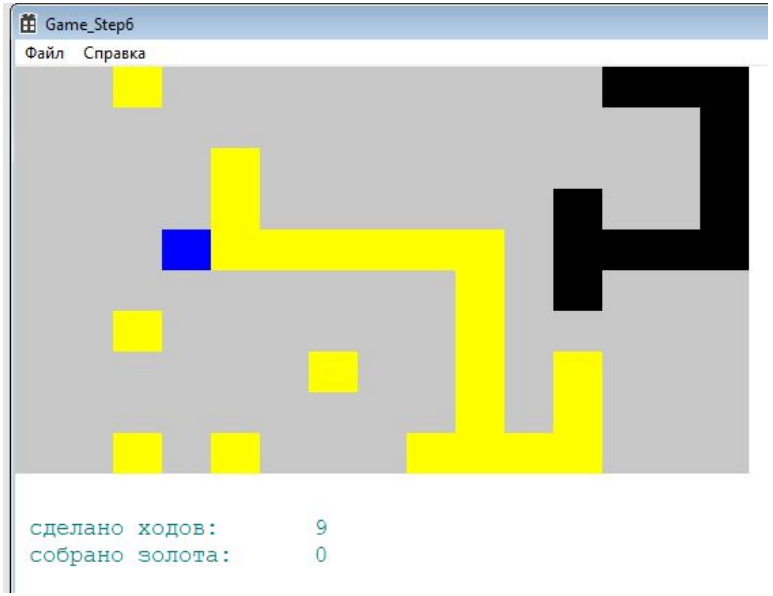
```
void doMidasHand(int i, int j) {
    if (a[i][j] == 2) { // 2 = стена
        a[i][j] = 1;    // 1 = золото
        if (i > 0) doMidasHand(i - 1, j);
        if (i < N - 1) doMidasHand(i + 1, j);
        if (j > 0) doMidasHand(i, j - 1);
        if (j < M - 1) doMidasHand(i, j + 1);
    }
}
```

Задача 8.2*

Выполнить трассировку в отладчике VS заливки небольшой стены при помощи добавленной функции

```
void doMidasHand(int i, int j)
```





Задача 9**

Реализовать бинарный поиск в отсортированном массиве.

Необходимо реализовать его двумя способами – итерационно и рекурсивно.

```c
#define NUM 10
int array[NUM] = {1, 3, 4, 8, 9, 10, 11, 15, 16, 18};

void printArray() {
    for (int i = 0; i < NUM; i++) {
        printf("%d ", array[i]);
    }
    printf("\n\n");
}
```

Задача 9** (2)

```
void main() {
    printArray();

    printf("contains(1) = %d\n", contains(1));
    printf("contains(2) = %d\n", contains(2));
    printf("contains(3) = %d\n", contains(3));
    printf("contains(4) = %d\n", contains(4));
    printf("contains(5) = %d\n", contains(5));
    printf("contains(6) = %d\n", contains(6));
    printf("contains(10) = %d\n", contains(10));
    printf("contains(15) = %d\n", contains(15));
    printf("\n\n");
    printf("containsR(1) = %d\n", containsR(0, NUM - 1, 1));
    printf("containsR(2) = %d\n", containsR(0, NUM - 1, 2));
    printf("containsR(3) = %d\n", containsR(0, NUM - 1, 3));
    printf("containsR(4) = %d\n", containsR(0, NUM - 1, 4));
    printf("containsR(5) = %d\n", containsR(0, NUM - 1, 5));
    printf("containsR(6) = %d\n", containsR(0, NUM - 1, 6));
    printf("containsR(10) = %d\n", containsR(0, NUM - 1, 10));
    printf("containsR(15) = %d\n", containsR(0, NUM - 1, 15));
}
```

Задача 9** (3)

```
// Бинарный поиск. Итерационная реализация
int contains(int value)
{
    int left = 0;
    int right = NUM - 1;

    while (left <= right) {
        int middle = (left + right) / 2;

        if (array[middle] == value) {
            return 1;
        }
        if (array[middle] < value) {
            left = middle + 1;
        } else { // if (array[middle] > value) {
            right = middle - 1;
        }
    }
    return 0;
}
```

Задача 9** (4)

```
// Бинарный поиск. Рекурсивная реализация
int containsR(int left, int right, int value)
{
    if (left > right) {
        return 0;
    }

    int middle = (left + right) / 2;

    if (array[middle] == value) {
        return 1;
    }

    if (array[middle] < value) {
        return containsR(middle + 1, right, value);
    } else { // if (array[middle] > value) {
        return containsR(left, middle - 1, value);
    }
}
```

Домашнее задание

1. Доделать задачи 1-9. (7** и 9** - необязательная задача)

2. Прорешать ВРУЧНУЮ задачи ЕГЭ - решение должно быть записано ОТ РУКИ в тетради/отчете. (задачи 4, 5, 6 и 7** (необязательная)).

3. Реализовать от двух до четырех рекурсивных функций не упомянутых в лекции и/или в лабораторной работе.
   Можно использовать задания по рекурсивным функциям из ЕГЭ, математические вычисления, рекурсивные реализации алгоритмов в игре. Засчитываются рекурсивные реализации обработки списков (лаб работа №12) и рекурсивные алгоритмы в вашей собственной игре.
   Фракталы – не засчитываются (это тема лабораторной работы №4).

4. Для двух рисунков, реализованных самостоятельно в рамках лаб работы №4 (тема «Фракталы») выполнить трассировку – в отладчике VS и вручную. Трассировку сделать на глубину трёх вызовов рекурсивных функций. В ручной трассировке необходимо