1 Массивы

1 Рекурсия

Повторим предыдущую тему: нарисуйте двоичное дерево, смотри рис. 1.

```
procedure btree(x, y, xstep, level: integer);
var
    newy, new_step: integer;
begin
    newy := y + 64;
    if (newy < getmaxy) and (xstep > 0) then
    begin
        new_step := xstep div 2;
        setcolor(level + 1);
        line(x, y, x - xstep, newy);
        line(x, y, x + xstep, newy);
        btree(x - xstep, newy, new_step, level + 1);
        btree(x + xstep, newy, new_step, level + 1)
    end
end;
```

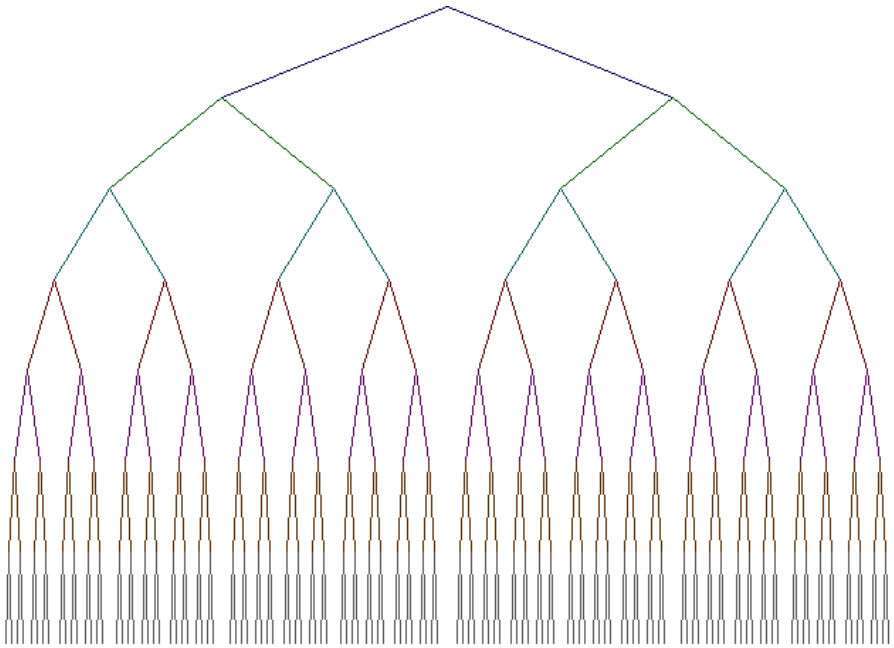

Рис. 1: Двоичное дерево.

2 Границы массива

Что произойдёт, если выйти за границы массива?

```
program boundaries;

var
    i: integer;
    x: integer;
    nums: array [1..10] of integer;
begin
{   nums[-34] := -32; }
    i := 0;
    x := 16;
    writeln('x = ', x);
    { now we do not touch variable j }
    nums[i] := -32;
    writeln('x = ', x);
    readln
end.
```

3 Случайные числа

Заполните массив из 24 целых чисел случайным образом от нуля до ста.

```
program fill_random;

```

```
var
    nums: array [1..24] of integer;
    i: integer;

begin
    for i := 1 to 24 do
        nums[i] := random(100);
    for i := 1 to 24 do
        writeln('nums [', i, '] = ', nums[i])
end.
```

4 Максимум

Найдите максимум в массиве из случайных чисел.

```
function maximum: integer;
var
    i, tmp: integer;
begin
    tmp := 0;
    for i := 1 to 24 do
        if tmp < nums[i] then
            tmp := nums[i];
    maximum := tmp;
end;
```

5 Минимум

Найдите минимум в массиве из случайных чисел.

6 Поиск элемента

В массиве из случайных чисел найдите позицию, в которой находится число 50. Если такой позиции нет, напишите -1. Выведите на экран, сколько раз приходится сравнивать элементы массива.

```
function find(a: integer): integer;
var
    i: integer;
begin
    find := -1;
    for i := 1 to 24 do
    begin
        if a = nums[i] then
        begin
            find := i;
            break
        end
    end
end;
```

7 Перестановка

Напишите процедуру, которая будет переставлять два элемента.

```
procedure perm(i, j: integer);
var
    tmp: integer;
begin
    tmp := nums[i];
    nums[i] := nums[j];
    nums[j] := tmp;
end;
```

8 Сортировка

Отсортируйте массив в порядке возрастания. Сколько перестановок потребовалось?

```
procedure buble;
var
    i, j: integer;
begin
    nperm := 0;
    for i := 1 to 24 do
        for j := i + 1 to 24 do
            if nums[j] < nums[i] then
                perm(i, j);
end;
```

9 Бинарный поиск

Сортируйте массив в порядке возрастания. При сортировке массива сохраните начальные позиции чисел в массиве.

```
function search(a: integer): integer;
label end_search;
var
    curr, step, ostep: integer;
    left, right: integer;
begin
    ncomp := 0;
    search := -1;
    curr := asize div 2;
    step := curr;
    repeat
        if nums[curr] = a then
        begin
            search := inds[curr];
            goto end_search
        end
        else
            if a < nums[curr] then
            begin
                curr := curr - step;
                if curr < 1 then
                    curr := 1;
            end
            else
            begin
                curr := curr + step;
                if curr > asize then
                    curr := asize;
            end;
        ostep := step;
        step := step div 2;
    until step <= 1;
    left := curr - ostep;
    if left < 1 then
        left := 1;
    right := curr + ostep;
    if right > asize then
        right := asize;
    for curr := left to right do
        if a = nums[curr] then
            search := inds[curr];
end_search: ;
end;
```

10 Стек

Стек — это структура данных в виде «стопки» однотипных переменных. Пользователь имеет доступ только до верхнего элемента. Напишите процедуру `push(x: integer)`, которая кладёт новый элемент `x` в «стопку», и функцию `pop: integer`, которая возвращает верхний элемент «стопки» и возвращает его значение.

```
program stack_program;

const ssize = 128;
var
    stack: array [1..ssize] of integer;
    sp: integer;

procedure init_stack;
begin
    sp := 0;
end;

procedure push(x: integer);
begin
    if sp < ssize then
    begin
        sp := sp + 1;
        stack[sp] := x;
    end
    else
    begin
        writeln('Stack overflow');
        exit;
    end
end;
```

```
26
27  function pop: integer;
28  begin
29      if sp >= 1 then
30      begin
31          pop := stack[sp];
32          sp := sp - 1;
33      end
34      else
35      begin
36          writeln('Stack␣underflow');
37          exit;
38      end
39  end;
40
41  var
42      i: integer;
43      tmp: integer;
44  begin
45      for i := 1 to 5 do begin
46          tmp := random(100);
47          writeln('push␣', tmp, '␣on␣stack');
48          push(tmp);
49      end;
50      for i := 1 to 5 do begin
51          writeln('pop␣␣', pop, '␣from␣stack');
52      end
53  end.
```