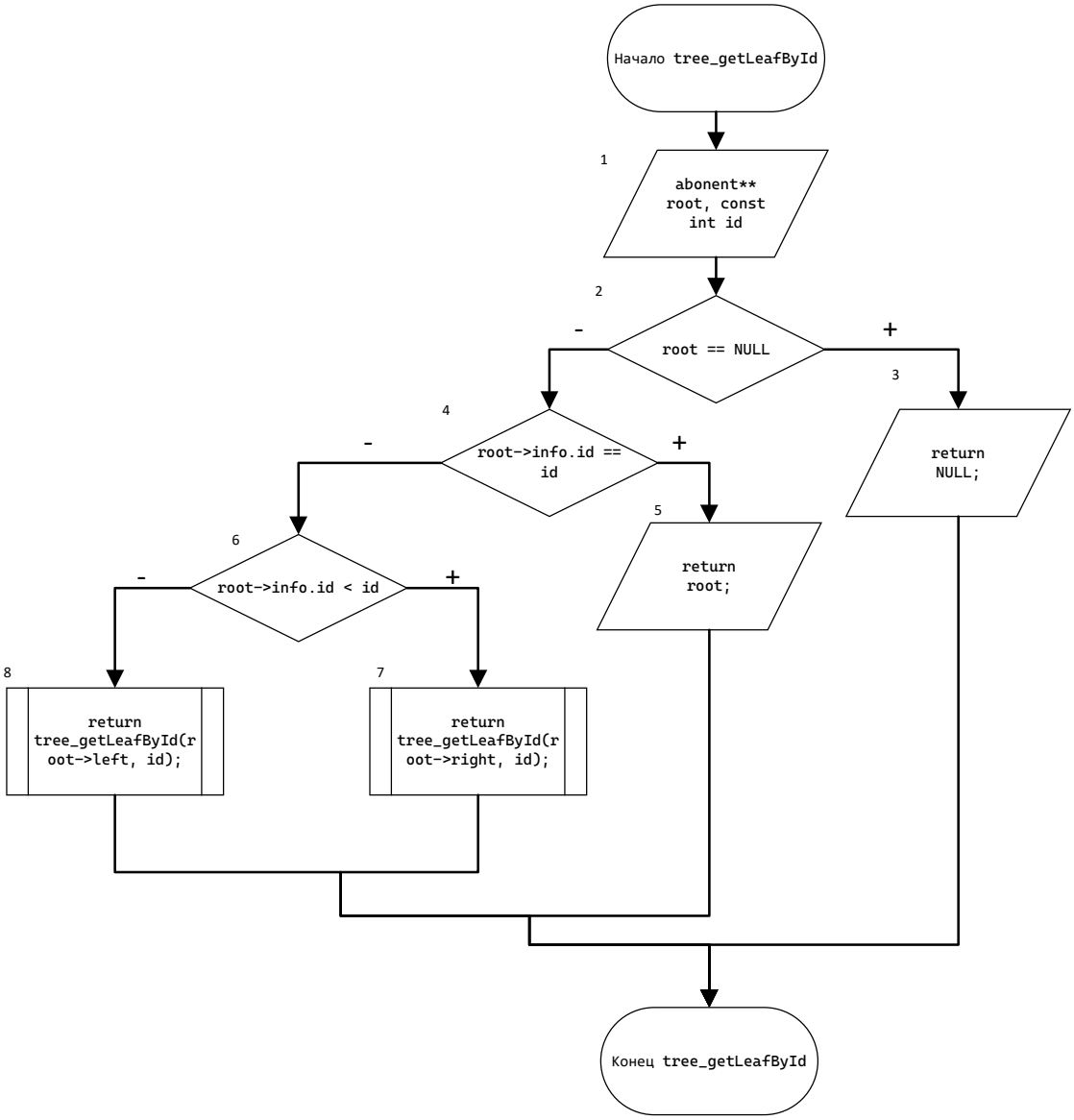
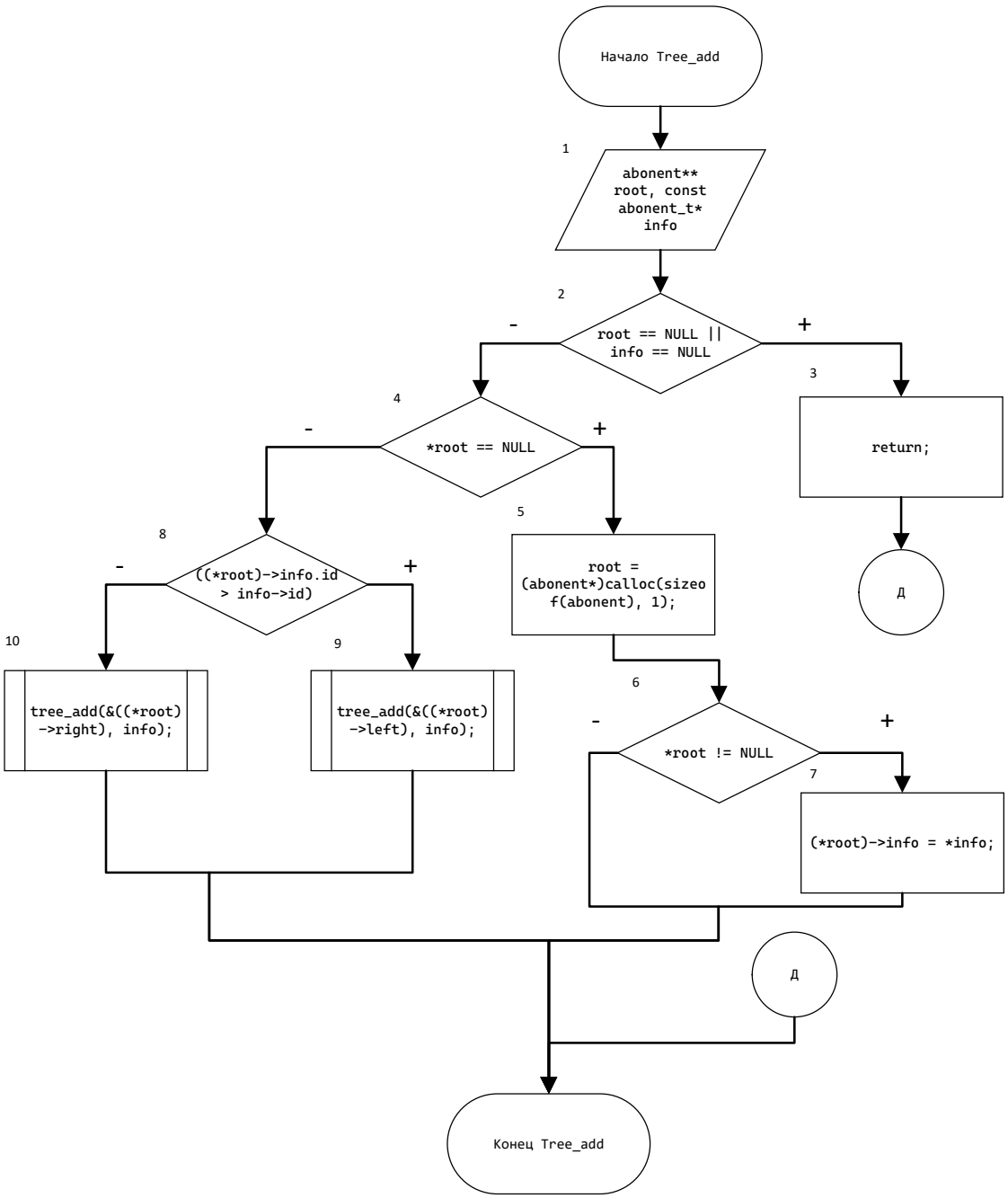
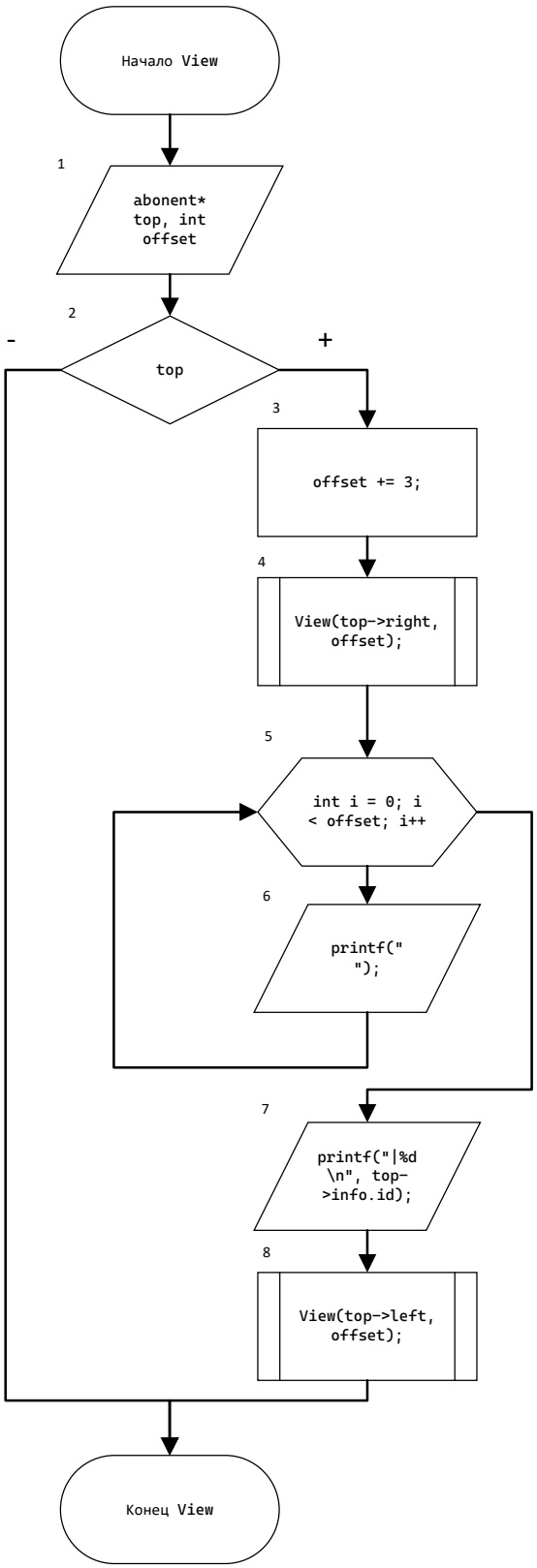
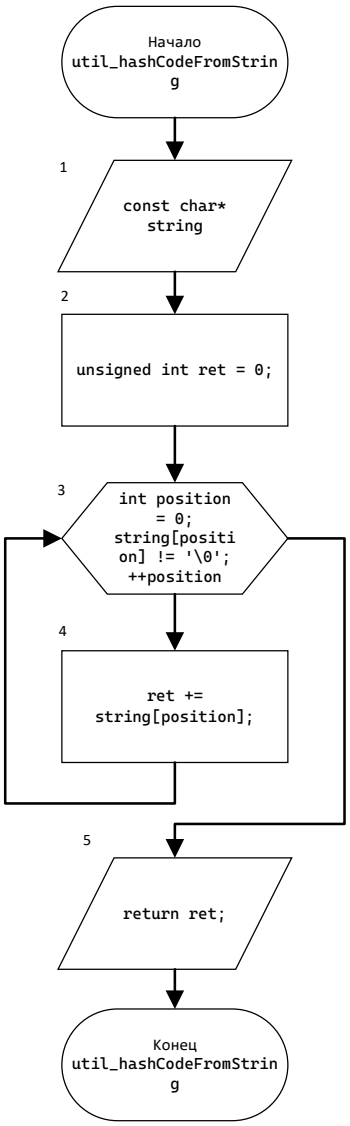
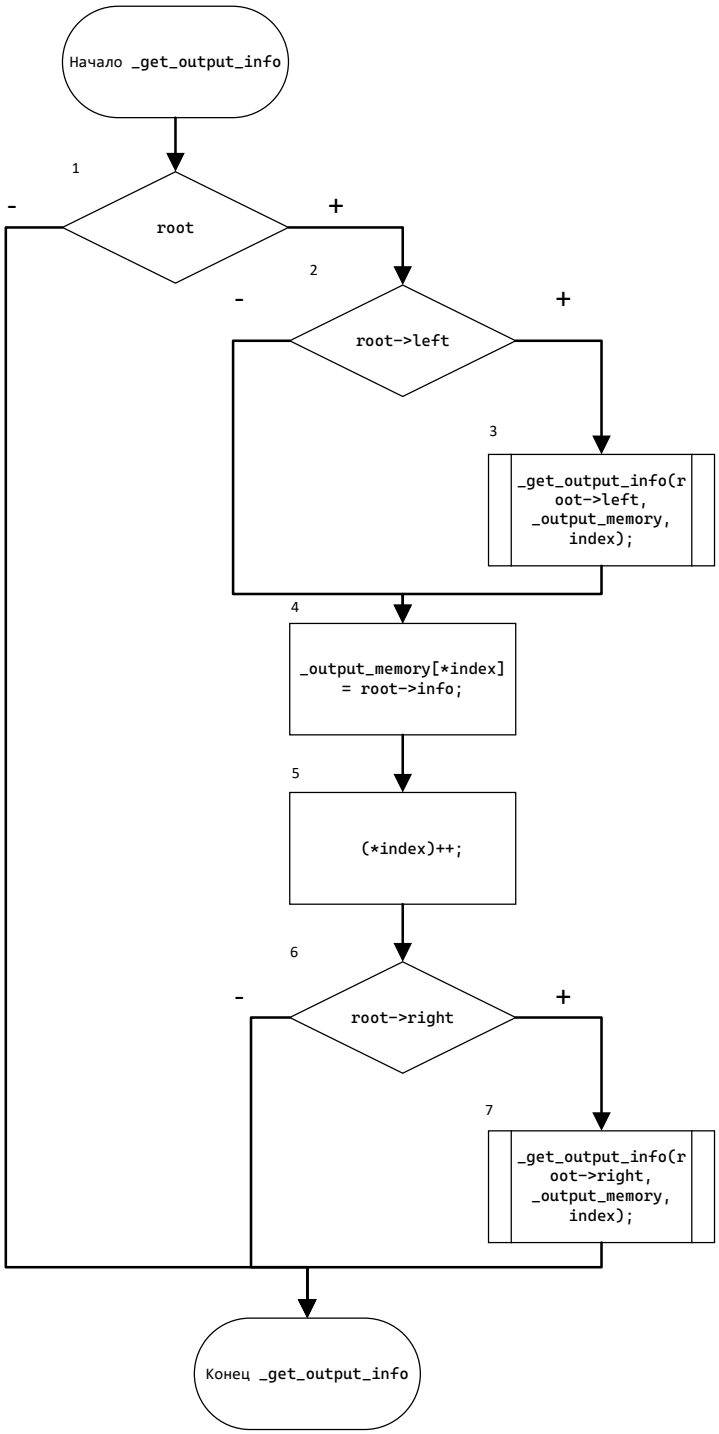
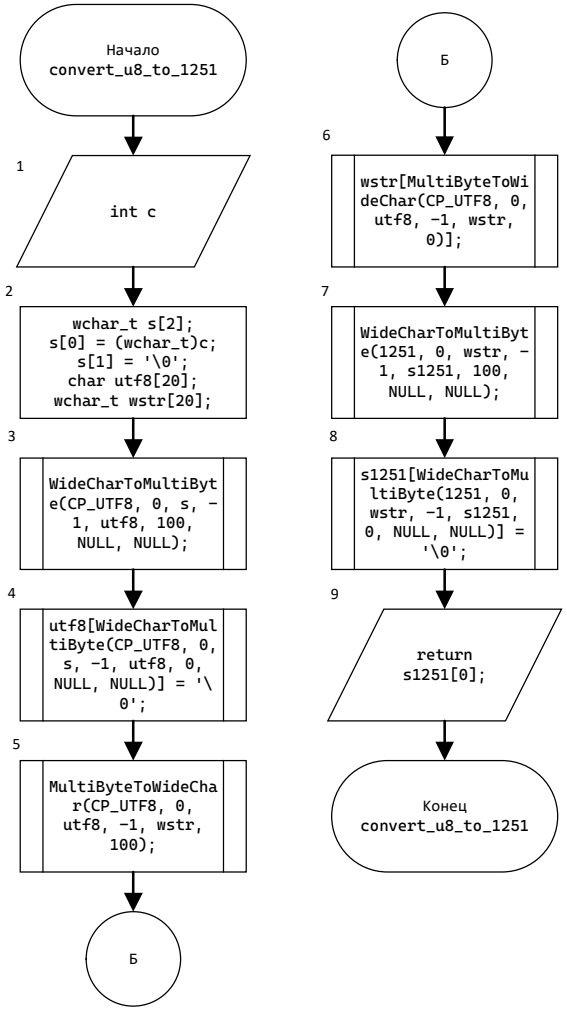

Начало tree_getLeafById
1
abonent** root, const int id
2
root == NULL
-
+
3
return NULL;
4
root->info.id == id
-
+
5
return root;
6
root->info.id < id
-
+
8
return tree_getLeafById(root->left, id);
7
return tree_getLeafById(root->right, id);
Конец tree_getLeafById

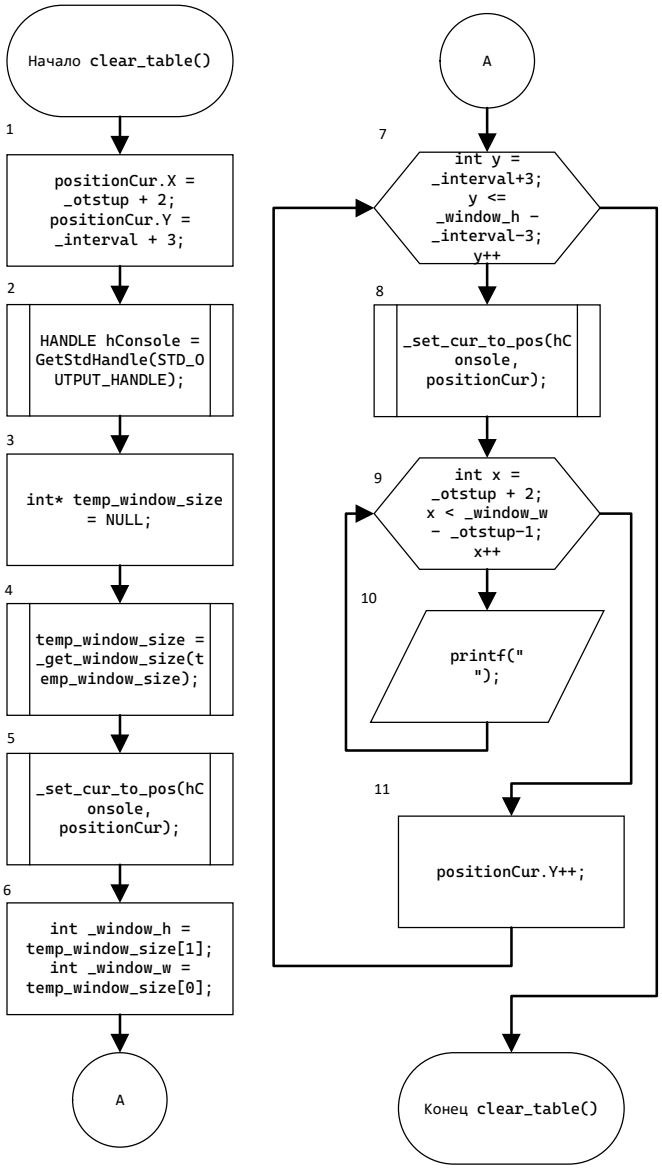

Начало Tree_add
1
abonent** root, const abonent_t* info
2
root == NULL || info == NULL
-
+
3
return;
Д
4
*root == NULL
-
+
5
root = (abonent*)calloc(sizeof(abonent), 1);
6
*root != NULL
-
+
7
(*root)->info = *info;
Д
8
((*root)->info.id > info->id)
-
+
9
tree_add(&((*root)->left), info);
10
tree_add(&((*root)->right), info);
Конец Tree_add

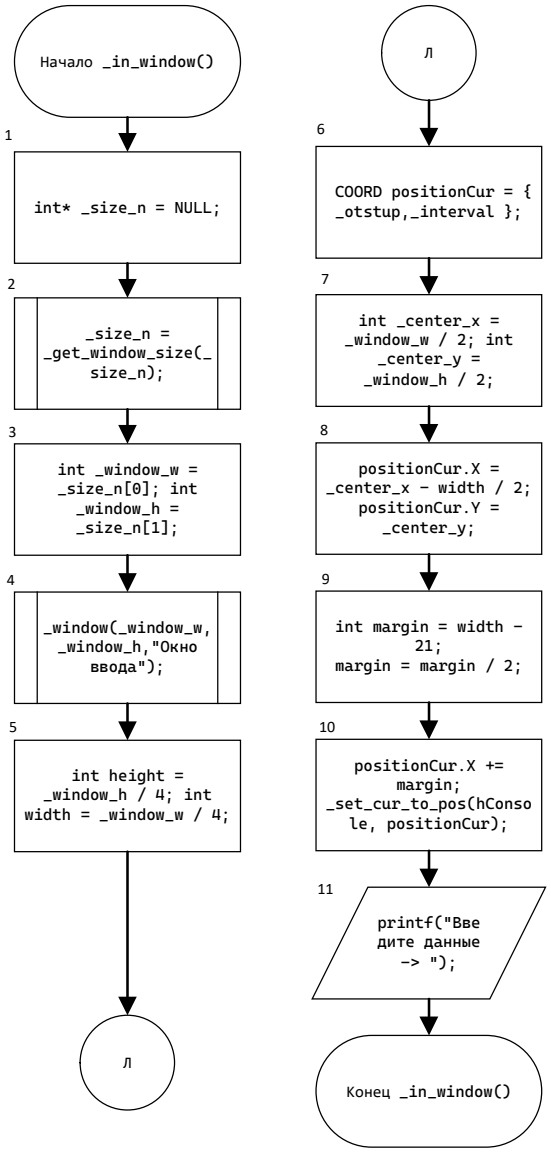

Начало View
1
abonent* top, int offset
2
top
-
+
3
offset += 3;
4
View(top->right, offset);
5
int i = 0; i < offset; i++
6
printf(" ");
7
printf("|%d \n", top->info.id);
8
View(top->left, offset);
Конец View

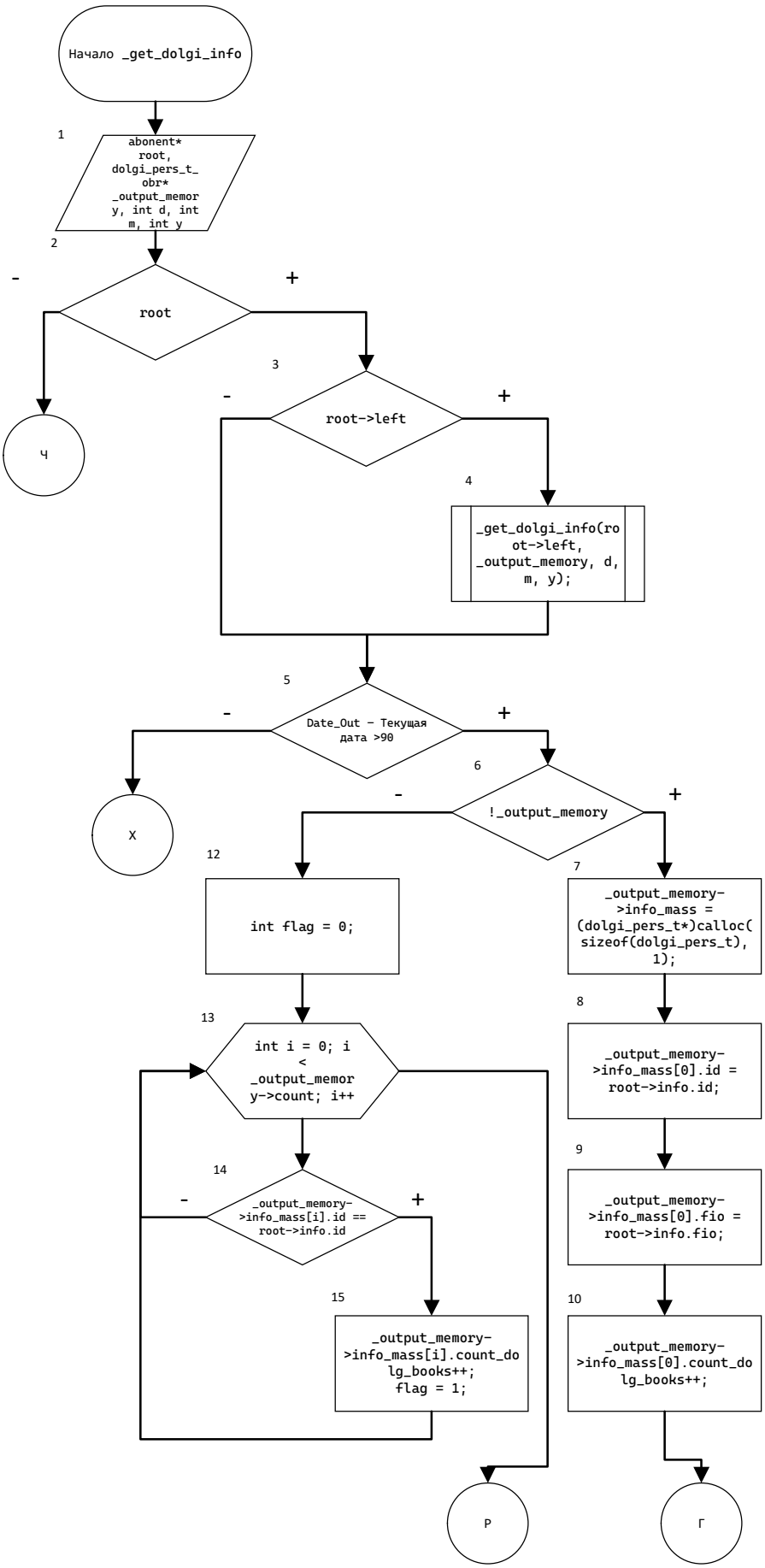

Начало util_hashCodeFromString
1
const char* string
2
unsigned int ret = 0;
3
int position = 0; string[position] != '\0'; ++position
4
ret += string[position];
5
return ret;
Конец util_hashCodeFromString

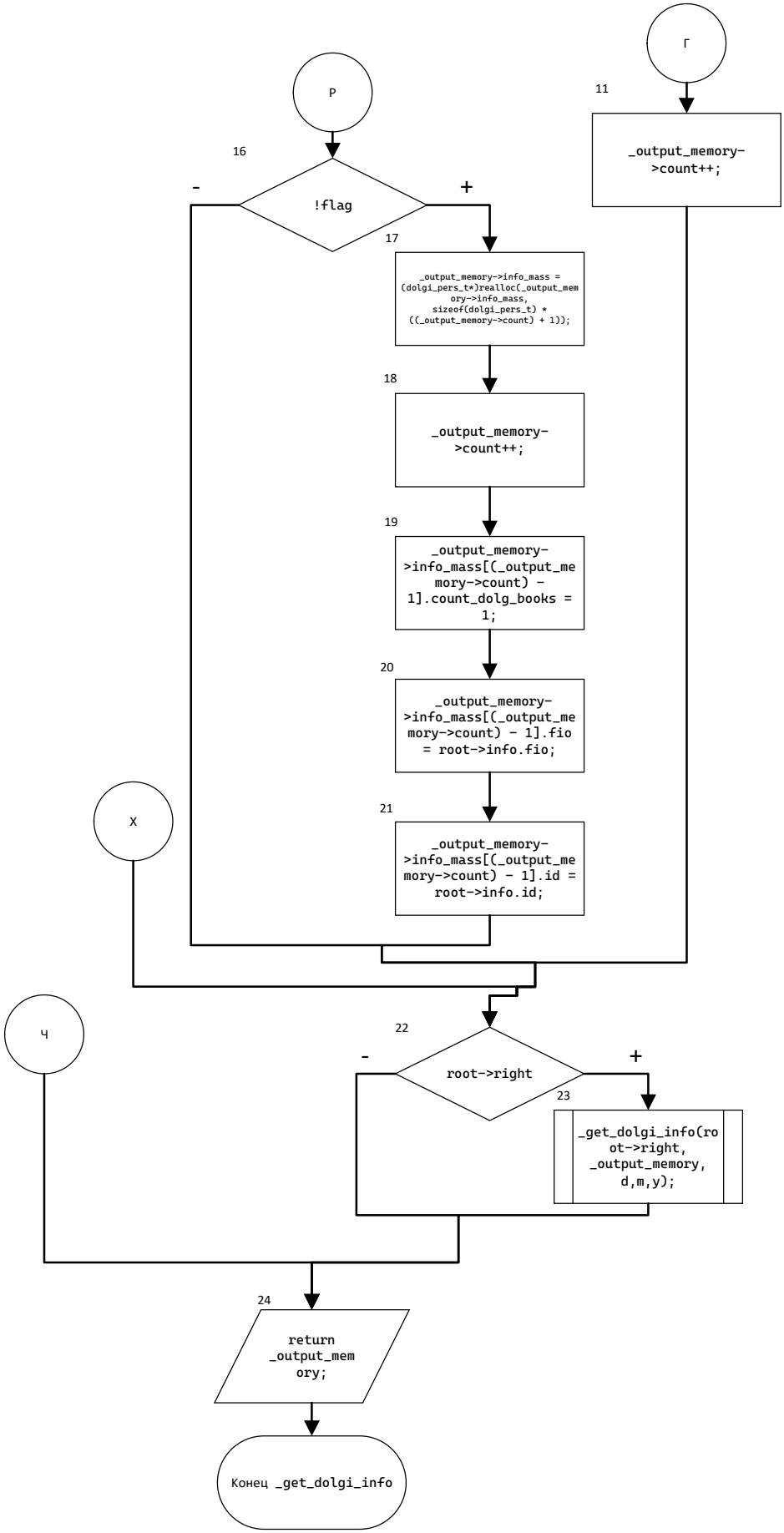

Начало _get_output_info
1
root
-
+
2
root->left
-
+
3
_get_output_info(root->left, _output_memory, index);
4
_output_memory[*index] = root->info;
5
(*index)++;
6
root->right
-
+
7
_get_output_info(root->right, _output_memory, index);
Конец _get_output_info

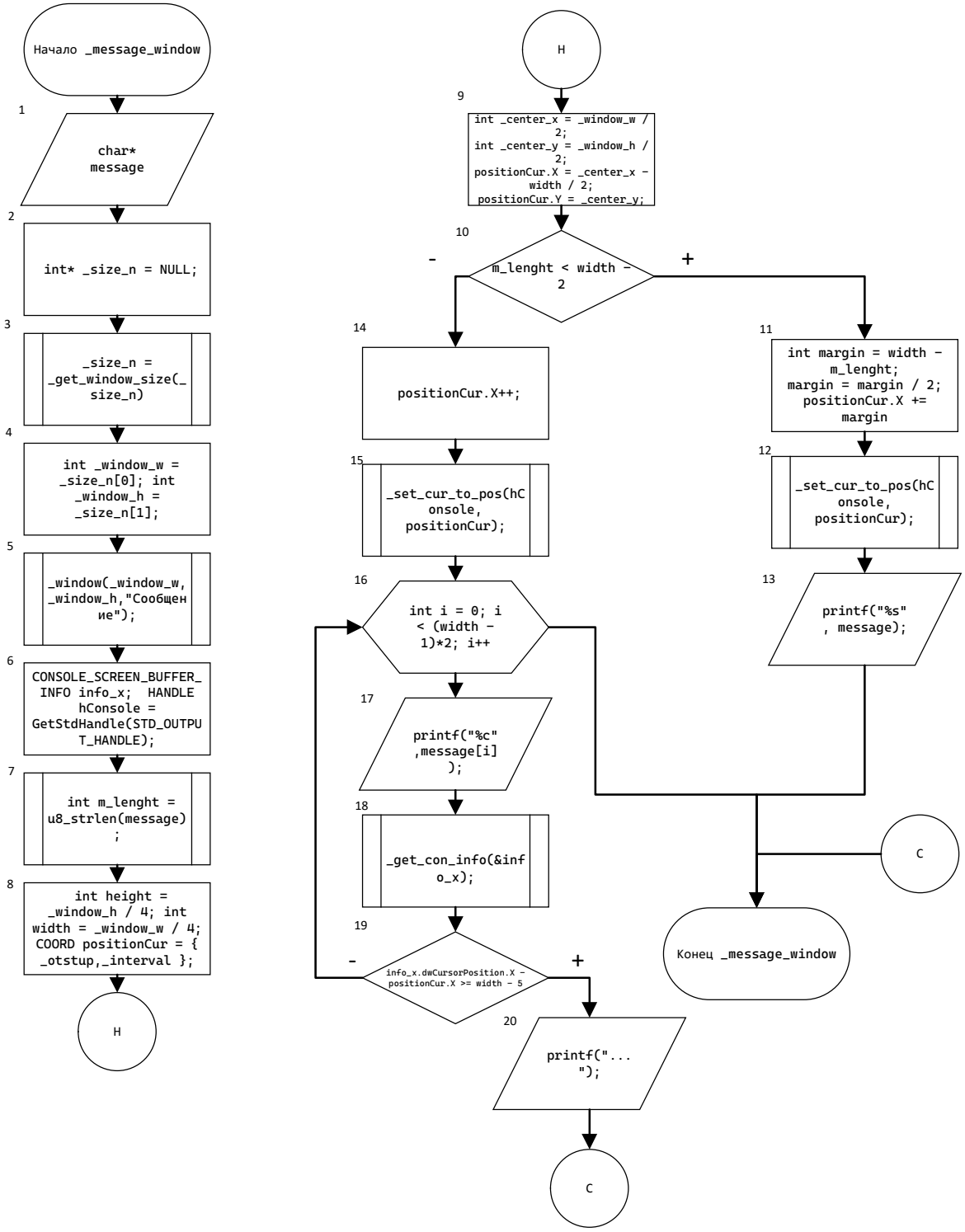

Начало convert_u8_to_1251
1
int c
2
wchar_t s[2];
s[0] = (wchar_t)c;
s[1] = '\0';
char utf8[20];
wchar_t wstr[20];
3
WideCharToMultiByte(CP_UTF8, 0, s, -1, utf8, 100, NULL, NULL);
4
utf8[WideCharToMultiByte(CP_UTF8, 0, s, -1, utf8, 0, NULL, NULL)] = '\0';
5
MultiByteToWideChar(CP_UTF8, 0, utf8, -1, wstr, 100);
Б
Б
6
wstr[MultiByteToWideChar(CP_UTF8, 0, utf8, -1, wstr, 0)];
7
WideCharToMultiByte(1251, 0, wstr, -1, s1251, 100, NULL, NULL);
8
s1251[WideCharToMultiByte(1251, 0, wstr, -1, s1251, 0, NULL, NULL)] = '\0';
9
return s1251[0];
Конец convert_u8_to_1251

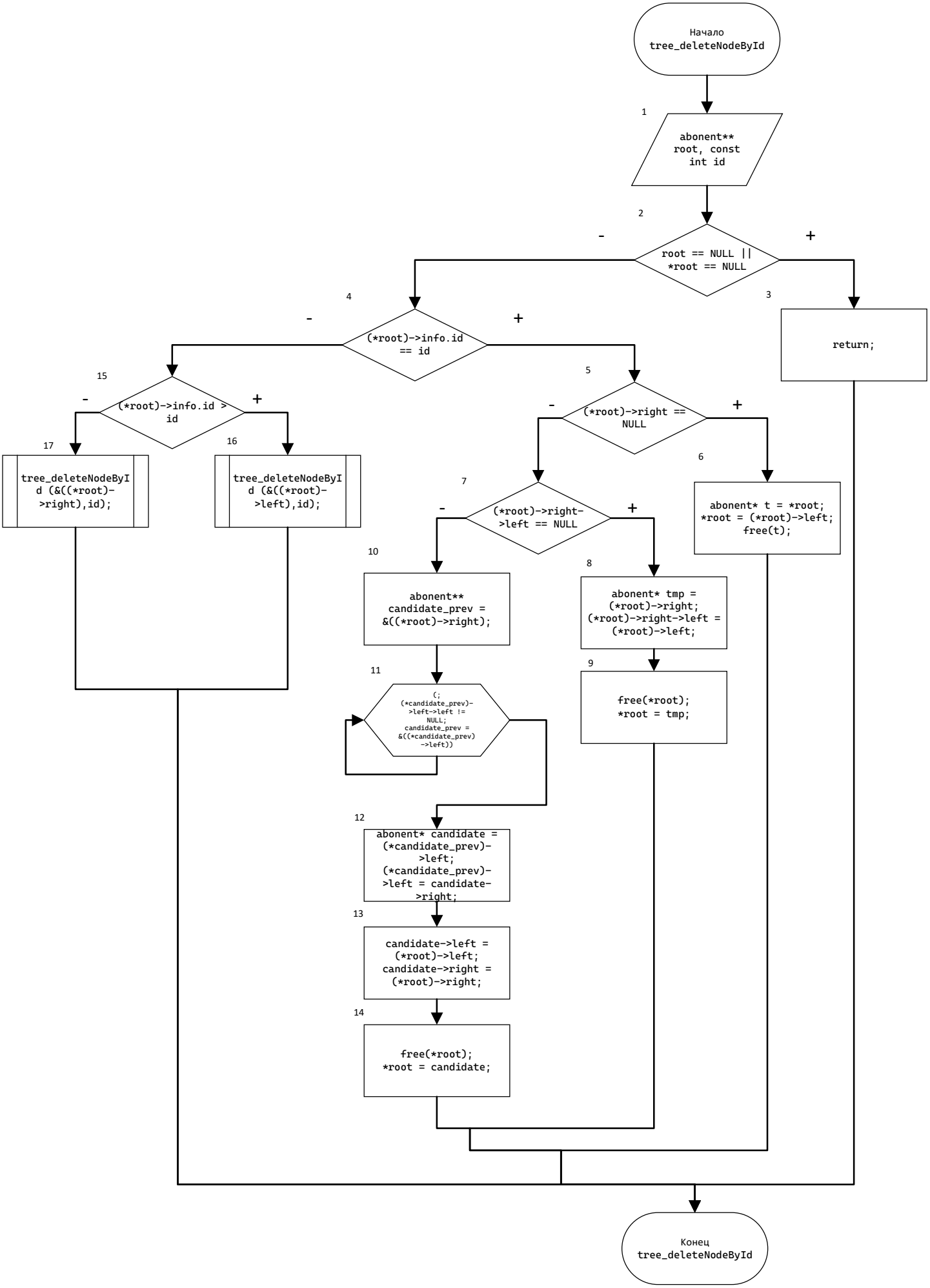

Начало clear_table()
1
positionCur.X = _otstup + 2;
positionCur.Y = _interval + 3;
2
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
3
int* temp_window_size = NULL;
4
temp_window_size = _get_window_size(temp_window_size);
5
_set_cur_to_pos(hConsole, positionCur);
6
int _window_h = temp_window_size[1];
int _window_w = temp_window_size[0];
А
А
7
int y = _interval+3; y <= _window_h - _interval-3; y++
8
_set_cur_to_pos(hConsole, positionCur);
9
int x = _otstup + 2; x < _window_w - _otstup-1; x++
10
printf(" ");
11
positionCur.Y++;
Конец clear_table()

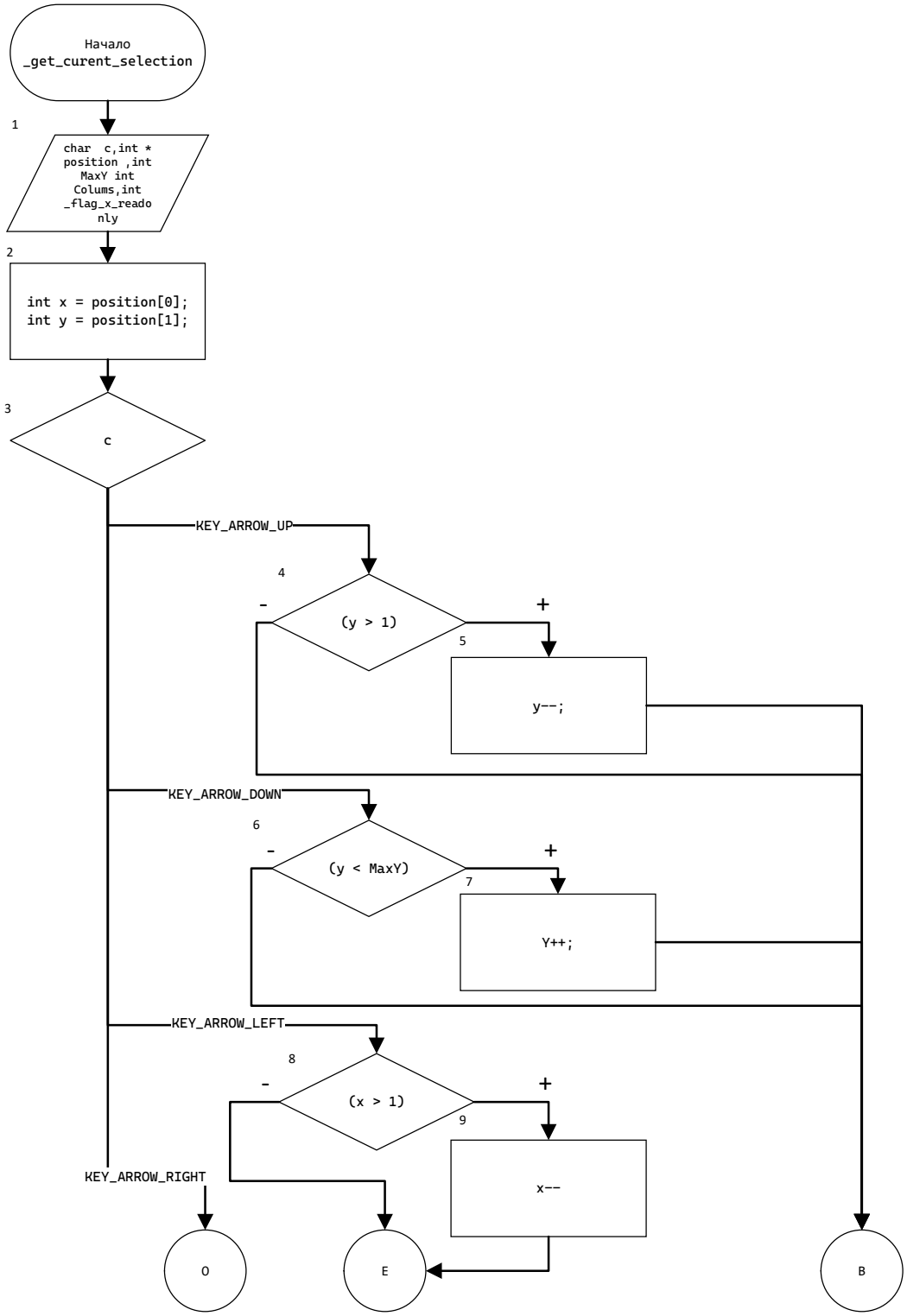

Начало _in_window()
1
int* _size_n = NULL;
2
_size_n = _get_window_size(_size_n);
3
int _window_w = _size_n[0]; int _window_h = _size_n[1];
4
_window(_window_w, _window_h,"Окно ввода");
5
int height = _window_h / 4; int width = _window_w / 4;
Л
Л
6
COORD positionCur = { _otstup,_interval };
7
int _center_x = _window_w / 2; int _center_y = _window_h / 2;
8
positionCur.X = _center_x - width / 2; positionCur.Y = _center_y;
9
int margin = width - 21; margin = margin / 2;
10
positionCur.X += margin; _set_cur_to_pos(hConsole, positionCur);
11
printf("Введите данные -> ");
Конец _in_window()

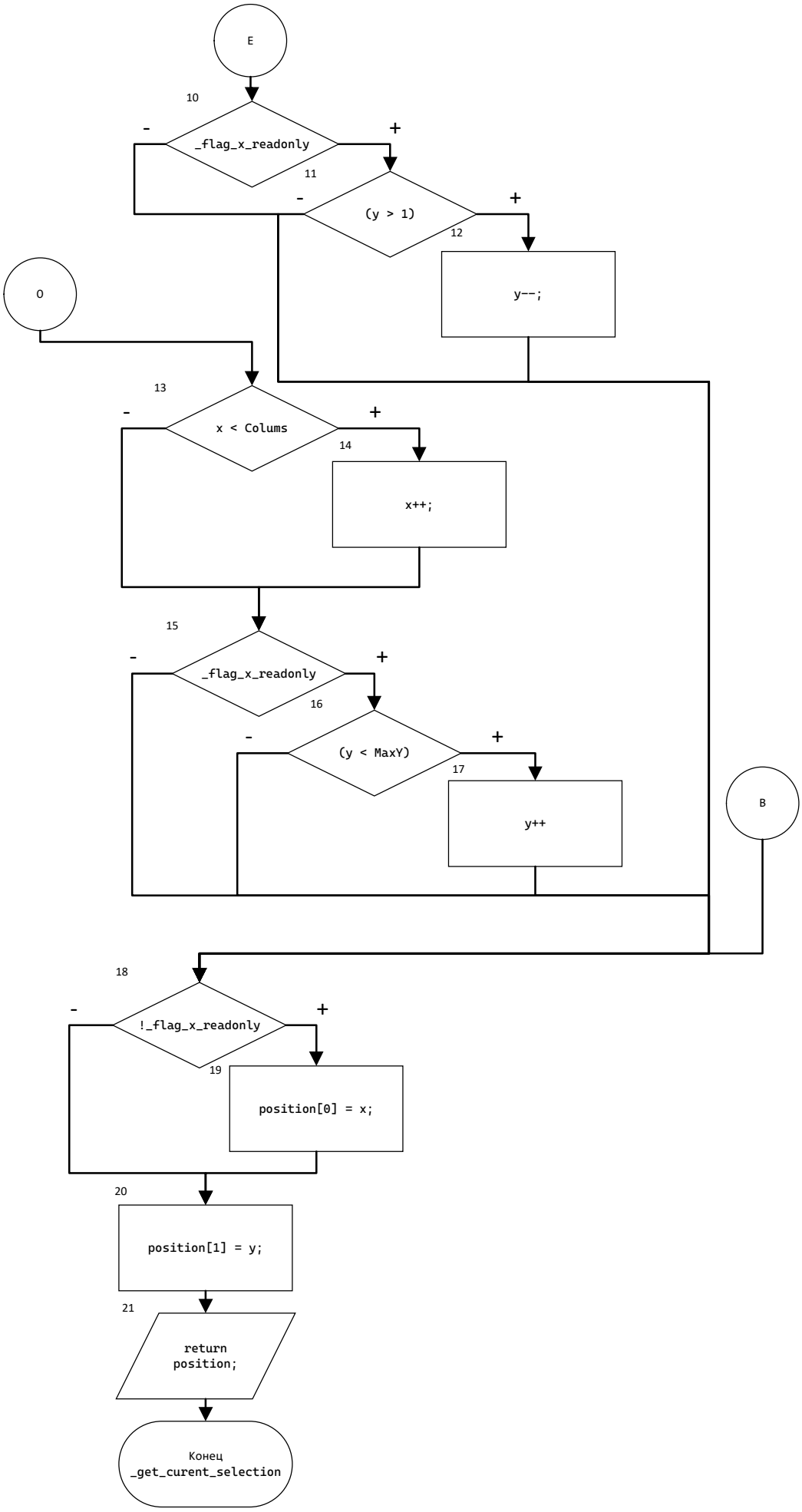

Начало _get_dolgi_info
1
abonent* root, dolgi_pers_t_ obr* _output_memory, int d, int m, int y
2
root
-
+
Ч
3
root->left
-
+
4
_get_dolgi_info(root->left, _output_memory, d, m, y);
5
Date_Out – Текущая дата >90
-
+
Х
6
!_output_memory
-
+
12
int flag = 0;
13
int i = 0; i < _output_memory->count; i++
14
_output_memory->info_mass[i].id == root->info.id
-
+
15
_output_memory->info_mass[i].count_dolg_books++; flag = 1;
Р
7
_output_memory->info_mass = (dolgi_pers_t*)calloc(sizeof(dolgi_pers_t), 1);
8
_output_memory->info_mass[0].id = root->info.id;
9
_output_memory->info_mass[0].fio = root->info.fio;
10
_output_memory->info_mass[0].count_dolg_books++;
Г


Г
11
_output_memory->count++;
Р
16
!flag
-
+
17
_output_memory->info_mass = (dolgi_pers_t*)realloc(_output_memory->info_mass, sizeof(dolgi_pers_t) * ((_output_memory->count) + 1));
18
_output_memory->count++;
19
_output_memory->info_mass[(_output_memory->count) - 1].count_dolg_books = 1;
20
_output_memory->info_mass[(_output_memory->count) - 1].fio = root->info.fio;
21
_output_memory->info_mass[(_output_memory->count) - 1].id = root->info.id;
Х
Ч
22
root->right
-
+
23
_get_dolgi_info(root->right, _output_memory, d,m,y);
24
return _output_memory;
Конец _get_dolgi_info


Начало _message_window
1
char* message
2
int* _size_n = NULL;
3
_size_n = _get_window_size(_size_n)
4
int _window_w = _size_n[0]; int _window_h = _size_n[1];
5
_window(_window_w, _window_h,"Сообщение");
6
CONSOLE_SCREEN_BUFFER_INFO info_x; HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
7
int m_lenght = u8_strlen(message);
8
int height = _window_h / 4; int width = _window_w / 4; COORD positionCur = { _otstup,_interval };
H
H
9
int _center_x = _window_w / 2; int _center_y = _window_h / 2; positionCur.X = _center_x - width / 2; positionCur.Y = _center_y;
10
m_lenght < width - 2
-
+
14
positionCur.X++;
15
_set_cur_to_pos(hConsole, positionCur);
16
int i = 0; i < (width - 1)*2; i++
17
printf("%c",message[i]);
18
_get_con_info(&info_x);
19
info_x.dwCursorPosition.X - positionCur.X >= width - 5
-
+
20
printf("...");
C
11
int margin = width - m_lenght; margin = margin / 2; positionCur.X += margin
12
_set_cur_to_pos(hConsole, positionCur);
13
printf("%s", message);
C
Конец _message_window


Начало
tree_deleteNodeById
1
abonent** root, const int id
2
root == NULL || *root == NULL
-
+
3
return;
4
(*root)->info.id == id
-
+
5
(*root)->right == NULL
-
+
6
abonent* t = *root; *root = (*root)->left; free(t);
7
(*root)->right->left == NULL
-
+
8
abonent* tmp = (*root)->right; (*root)->right->left = (*root)->left;
9
free(*root); *root = tmp;
10
abonent** candidate_prev = &((*root)->right);
11
(; (*candidate_prev)->left->left != NULL; candidate_prev = &((*candidate_prev)->left))
12
abonent* candidate = (*candidate_prev)->left; (*candidate_prev)->left = candidate->right;
13
candidate->left = (*root)->left; candidate->right = (*root)->right;
14
free(*root); *root = candidate;
15
(*root)->info.id > id
-
+
16
tree_deleteNodeById (&((*root)->left),id);
17
tree_deleteNodeById (&((*root)->right),id);
Конец
tree_deleteNodeById


Начало
_get_curent_selection
1
char c,int *
position ,int
MaxY int
Colums,int
_flag_x_reado
nly
2
int x = position[0];
int y = position[1];
3
c
KEY_ARROW_UP
4
-
(y > 1)
+
5
y--;
KEY_ARROW_DOWN
6
-
(y < MaxY)
+
7
Y++;
KEY_ARROW_LEFT
8
-
(x > 1)
+
9
x--
KEY_ARROW_RIGHT
O
E
B


E
10
_flag_x_readonly
-
+
11
(y > 1)
-
+
12
y--;
O
13
x < Colums
-
+
14
x++;
15
_flag_x_readonly
-
+
16
(y < MaxY)
-
+
17
y++
B
18
!_flag_x_readonly
-
+
19
position[0] = x;
20
position[1] = y;
21
return
position;
Конец
_get_curent_selection