

## Файл list.h

```
#ifndef _LIST_H
#define _LIST_H

// Структура, реализующая ячейку списка, в котором каждый элемент —
// целое число.
typedef struct IntList {
    int value;
    struct IntList *next;
} IntList;

// Создать список.
IntList *int_list_create();
// Уничтожить список, освободив память.
void int_list_destroy(IntList *list);

// Вернуть количество элементов списка.
int int_list_length(IntList *list);
// Добавить новый элемент в начало списка.
IntList *int_list_push(IntList *list, int newElem);
// Удалить начальный элемент списка
IntList *int_list_remove_0(IntList *list);
// Вернуть n-ое значение из списка, элементы индексируются с 0.
int int_list_nth_value(IntList *list, int n);

#endif // _LIST_H
```

## Файл main.c

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

int main()
{
    int a, value, len, i;
    IntList *list;
    list = int_list_create();

    while (1) {
        printf("1_Добавить_значение_в_стек\n2_Распечатать_значение_из_головы_стека\n"
            "3_Распечатать_все_значения_стека\n4_Удалить_вершину_стека\n"
            "5_Выход\n\n" "Выберите_пункт_меню:_\n");
        while (scanf("%d", &a) != 1 && (a < 1 || a > 5));
        switch (a) {
            case 1:
                while (scanf("%d", &value) != 1);
                list = int_list_push(list, value);
                break;
            case 2:
                printf("Вершина_стека:_%d\n", int_list_nth_value(list, 0));
                break;
            case 3:
                len = int_list_length(list);
                for (i = 0; i < len; i++)
                    printf("%d_", int_list_nth_value(list, i));
                printf("\n");
                break;
            case 4:
                list = int_list_remove_0(list);
                break;
            case 5:
                int_list_destroy(list);
                return 0;
        }
    }
}
```

```

        break;
    default:
        printf
            ("Странно. Этот кусок кода не должен выполняться, \n"
             "значит где-то в программе ошибка. Завершаем работу.\n");
        return 1;
        break;
    }
}

```

#### Файл list.c

```

#include "list.h"
#include <stdlib.h>

IntList *int_list_create()
{
    return NULL;
}

void int_list_destroy(IntList *list)
{
    IntList *current, *previous;
    previous = list;
    for (current = list->next; current != NULL; current = current->next) {
        free(previous);
        previous = current;
    }
    free(previous);
}

int int_list_length(IntList *list)
{
    IntList *current;
    int length = 0;
    for (current = list; current != NULL; current = current->next)
        length++;
    return length;
}

IntList *int_list_push(IntList *list, int newElem)
{
    IntList *newCell;
    newCell = (IntList *) malloc(sizeof(IntList));
    newCell->value = newElem;
    newCell->next = list;
    return newCell;
}

IntList *int_list_remove_0(IntList *list)
{
    IntList *tail = list->next;
    free(list);
    return tail;
}

int int_list_nth_value(IntList *list, int n)
{
    IntList *current = list;
    while(n-- > 0)
        current = current->next;
    return current->value;
}

```