

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем

ОТЧЕТ
по практической работе 2
по дисциплине «Программирование»

Выполнил:
студент гр. ИВ-921
«22» марта 2020 г.

/Черемисин И.И./

Проверил:
старший преподаватель Кафедры ВС
«__» _____ 2020 г.

/Фульман В.О./

Оценка « _____ »

Новосибирск 2020

ОГЛАВЛЕНИЕ

ЗАДАНИЕ.....	3
ВЫПОЛНЕНИЕ РАБОТЫ.....	4
ПРИЛОЖЕНИЕ.....	6

ЗАДАНИЕ

Реализовать тип данных «Динамический массив целых чисел» — `IntVector` и основные функции для работы с ним. Разработать тестовое приложение для демонстрации реализованных функций.

Рекомендуемая структура проекта:

```
.
|-- Makefile
`-- src
    |-- IntVector.c
    |-- IntVector.h
    `-- main.c
```

Advanced: в функциях получения элемента по индексу и установки значения заданного элемента предусмотреть обработку выхода за границы массива. Продумать сигнатуру функции, позволяющую обработать ошибку в клиентском коде.

Требования к работе

1. Должны обрабатываться ошибки выделения памяти.
2. Не должно быть утечек памяти.
3. При тестировании приложения необходимо проверить граничные случаи. Например, работоспособность операции добавления элемента после уменьшения размера массива до нуля.

ВЫПОЛНЕНИЕ РАБОТЫ

1 шаг. В файле `IntVector.c` (файл реализации):

Объявил функцию `int_vector_new`, в котором реализовал массив нулевого размера.

Затем с помощью `int_vector_free` освобождаем память, выделенную для вектора `v`.

В функции `int_vector_get_item` задал элемент `index`, потом в функции `int_vector_set_item` присваиваю элементу под номером `index` значение `item`.

В `int_vector_get_size` выводит размер вектора, в `int_vector_get_capacity` выводит емкость вектора.

Затем объявляю функцию `int_vector_push_back`, которая добавляет элемент в конец массива и при необходимости увеличивает емкость массива. Функция

`int_vector_pop_back` удаляет последний элемент из массива (Если размер массива 0, то эффекта не будет), после задал функцию

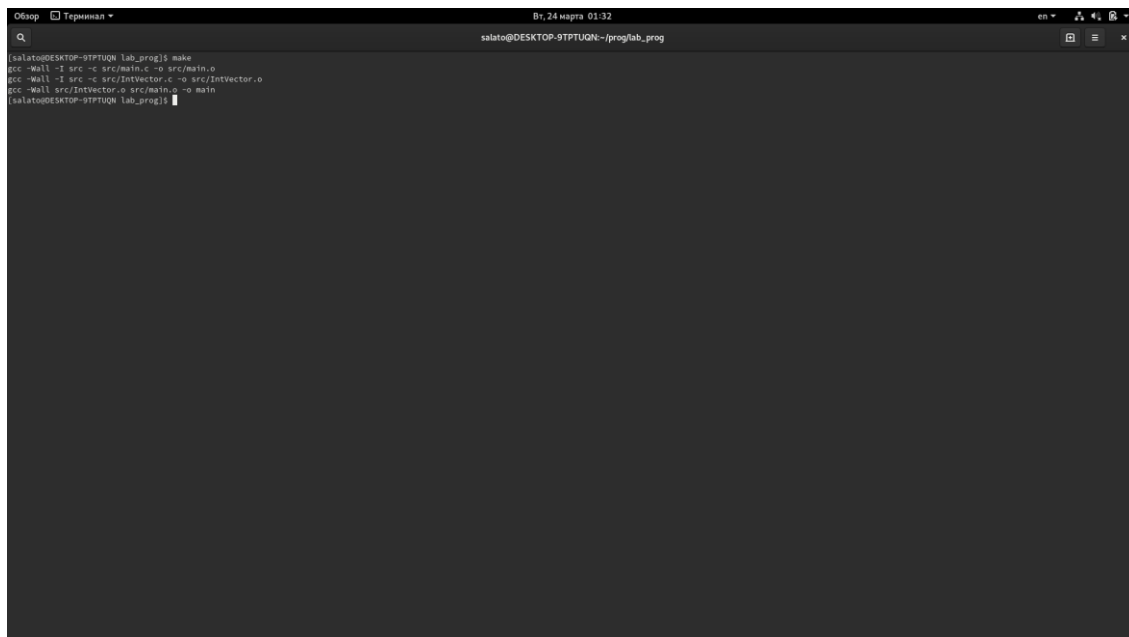
`int_vector_shrink_to_fit`, которая уменьшает емкость массива до его размера.

Для изменения массива объявил функцию `int_vector_resize` (Если новый размер массива больше исходного, то добавленные элементы заполняются нулями, если же новый размер массива меньше исходного, то перевыделение памяти не происходит)

2 шаг. Файл `IntVector.h` (заголовочный файл): хранит в себе функции из файла реализации

3 шаг. Файл `main.c` (тестовый кейс)

Компиляция программы



```
salato@DESKTOP-9TPTUQN:~/prog/lab_prog$ make
gcc -Wall -I src -c src/main.c -o src/main.o
gcc -Wall -I src -c src/IntVector.c -o src/IntVector.o
gcc -Wall src/IntVector.o src/main.o -o main
salato@DESKTOP-9TPTUQN:~/prog/lab_prog$
```

Запуск программы

```
Обзор Терминал
By 24 марта 01:42
salato@DESKTOP-9TPTUQN:~/prog/lab_prog

(salato@DESKTOP-9TPTUQN lab_prog) make
gcc -Wall -I src -c src/main.c -o src/main.o
gcc -Wall -I src -c src/IntVector.c -o src/IntVector.o
gcc -Wall src/IntVector.o src/main.o -o main
(salato@DESKTOP-9TPTUQN lab_prog) ./main
A = 0 1 2 3 0
B = 0 1 2 3 0
S 5 5
B = 0 1 2 3 5
S 10 6
B = get item = 9
capacity = 10
size = 5
capacity = 5
20
0 1 2 3 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
20
Segmentation fault (core dumped)
(salato@DESKTOP-9TPTUQN lab_prog)
```

ПРИЛОЖЕНИЕ

IntVector.c

```
1 #include <stdlib.h>
2 #include <string.h>
3 #include "IntVector.h"
4
5 IntVector *int_vector_new(size_t initial_capacity) /*создание массива нулевого
6 размера*/
7 {
8     IntVector *a;
9     a = (IntVector *)malloc(sizeof(IntVector)); /*выделение участка динамической
10 памяти*/
11     if (!a)
12     {
13         return NULL;
14     }
15     a->arr = (int *)malloc(initial_capacity * sizeof(int)); /*указатель на участок
16 памяти,
17 в котором хранятся элементы. sizeof(int) - первого элемента массива*/
18     if (!a->arr)
19     {
20         free(a); /*освобождение памяти, которая выделена с использованием одной из
21 функции:
22 malloc(), realloc() или calloc()*/
23         return NULL;
24     }
25     a->capacity = initial_capacity; /*емкость вектора*/
26     a->size = 0; /*размер вектора*/
27     return a;
28 }
29
30 IntVector *int_vector_copy(const IntVector *v) /*указатель на копию веткора v,
31 если не удалось выделять паммать - NULL*/
32 {
33     IntVector *a = int_vector_new(v->capacity);
34     if (!a)
35     {
36         return NULL;
37     }
38     a->arr = memcpy(a->arr, v->arr, (sizeof(int) * (v->size))); /*копирование одной
39 области памяти в другую*/
40     if (!a->arr)
41     {
42         free(a);
43         return NULL;
44     }
45     a->size = v->size;
```

```

46     return a;
47 }
48
49 void int_vector_free(IntVector *v) /*освобождает память, выделенную для вектора v*/
50 {
51     free(v->arr);
52     free(v);
53 }
54
55 int int_vector_get_item(const IntVector *v, size_t index)
56 {
57     return v->arr[index]; /*элемент под номером index*/
58 }
59
60 void int_vector_set_item(IntVector *v, size_t index, int item)
61 {
62     v->arr[index] = item; /*Присваивает элементу под номером index значение item*/
63 }
64
65 size_t int_vector_get_size(const IntVector *v)
66 {
67     return v->size; /*размер вектора*/
68 }
69
70 size_t int_vector_get_capacity(const IntVector *v)
71 {
72     return v->capacity; /*емкость вектора*/
73 }
74
75 int int_vector_push_back(IntVector *v, int item) /*Добавляет элемент в конец массива.
76 При необходимости увеличивает емкость массива.*/
77 {
78     if (v->size == v->capacity) /*в случае ошибки*/
79     {
80         int *array = realloc(v->arr, (v->capacity * sizeof(int) * 2));
81         if (!array)
82         {
83             return -1;
84         }
85         v->arr = array;
86         v->capacity *= 2;
87     }
88     v->arr[v->size] = item;
89     v->size++;
90     return 0;
91 }
92
93 void int_vector_pop_back(IntVector *v) /*Удаляет последний элемент из массива.
94 Нет эффекта, если размер массива равен 0.*/
95 {

```

```

96     if (v->size != 0 && v->capacity != 0)
97     {
98         v->arr[v->size - 1] = 0;
99         v->size--;
100    }
101}
102
103int int_vector_shrink_to_fit(IntVector *v)/*Уменьшает емкость массива до его размера*/
104{
105    int *array = realloc(v->arr, (v->size * sizeof(int)));
106    if (!array)
107    {
108        return -1;
109    }
110    v->arr = array;
111    v->capacity = v->size;
112    return 0;
113}
114
115int int_vector_resize(IntVector *v, size_t new_size)/*Изменяет размер массива*/
116{
117    if (v->size < new_size)/*Если новый размер массива больше исходного,
118то добавленные элементы заполняются нулями*/
119    {
120        int *array = realloc(v->arr, (new_size * sizeof(int)));
121        if (!array)
122        {
123            return -1;
124        }
125        v->arr = array;
126        for (size_t i = v->size; i < new_size; i++)
127            v->arr[i] = 0;
128        v->size = new_size;
129        v->capacity = new_size;
130    }
131    if (v->size > new_size)/*Если новый размер массива меньше исходного,
132то перевыделение памяти не происходит*/
133    {
134        v->size = new_size;
135    }
136    return 0;
137}
138
139int int_vector_reserve(IntVector *v, size_t new_capacity)/*Изменить емкость массива*/
140{
141    if (v->capacity >= new_capacity)/*Если новая емкость меньше либо равна исходной -
142нет эффекта*/
143    {
144        return 0;
145    }

```



```
146     int *array = realloc(v->arr, (new_capacity * sizeof(int)));
147     if (!array)
148     {
149         return -1;
150     }
151     v->arr = array;
152     v->capacity = new_capacity;
153     return 0;
154 }
```

IntVector.h

```
1 #ifndef INTVECTOR_H
2 #define INTVECTOR_H
3
4 typedef struct
5 {
6     int *arr;
7     size_t size;
8     size_t capacity;
9 } IntVector;
10 IntVector *int_vector_new(size_t initial_capacity);
11 IntVector *int_vector_copy(const IntVector *v);
12 void int_vector_free(IntVector *v);
13 int int_vector_get_item(const IntVector *v, size_t index);
14 void int_vector_set_item(IntVector *v, size_t index, int item);
15 size_t int_vector_get_size(const IntVector *v);
16 size_t int_vector_get_capacity(const IntVector *v);
17 int int_vector_push_back(IntVector *v, int item);
18 void int_vector_pop_back(IntVector *v);
19 int int_vector_shrink_to_fit(IntVector *v);
20 int int_vector_resize(IntVector *v, size_t new_size);
21 int int_vector_reserve(IntVector *v, size_t new_capacity);
22 #endif
23 /*#ifndef и #endif - Директивы условной компиляции препроцессора, позволяют
    компилировать часть программы*/
```

main.c

```
1 #include <stdio.h>
2 #include "IntVector.h"
3
4 int main()
5 {
6
7     IntVector *a, *b;
8     a = int_vector_new(5);
9     for (int i = 0; i < 4; i++)
10     {
11         a->size++;
12         a->arr[i] = i;
13     }
14     printf("A = ");
15     for (int i = 0; i < 5; i++)
16     {
17         printf("%d ", a->arr[i]);
18     }
19     b = int_vector_copy(a);
20     printf("\nB = ");
21     for (int i = 0; i < 5; i++)
22     {
23         printf("%d ", b->arr[i]);
24     }
25     int_vector_push_back(b, 5);
26     printf("\n%d %zd %zd\n", b->arr[4], int_vector_get_capacity(b),
27 int_vector_get_size(b));
28     printf("\nB = ");
29     for (int i = 0; i < 5; i++)
30     {
31         printf("%d ", b->arr[i]);
32     }
33     int_vector_push_back(b, 5);
34     printf("\n%d %zd %zd\n", b->arr[5], int_vector_get_capacity(b),
35 int_vector_get_size(b));
36     printf("\nB = ");
37     int_vector_set_item(b, 0, 9);
38     printf("get item = %d\n", int_vector_get_item(b, 0));
39     int_vector_pop_back(b);
40     printf("capacity = %zd\nsize = %zd\n", int_vector_get_capacity(b),
41 int_vector_get_size(b));
42     int_vector_shrink_to_fit(b);
43     printf("capacity = %zd\n", int_vector_get_capacity(b));
44     int_vector_resize(b, 20);
45     printf("%zd\n", int_vector_get_size(b));
```

```
46     int_vector_reserve(b, 20);
47     for (int i = 0; i < 20; i++)
48     {
49         printf("%d ", b->arr[i]);
50     }
51     printf("\n%zd\n", int_vector_get_capacity(b));
52     int_vector_free(b);
53     int_vector_free(a);
54     for (int i = 0; i < 20; i++)
55     {
56         printf("%d ", b->arr[i]);
57     }

    return 0;
}
```

Makefile

```
1 all: main
2
3 main: src/main.o src/IntVector.o /*<цель>: [зависимость]*/
4     gcc -Wall src/IntVector.o src/main.o -o main
5
6 src/main.o: src/main.c src/IntVector.h
7     gcc -Wall -I src -c src/main.c -o src/main.o
8
9 src/IntVector.o: src/IntVector.c
10    gcc -Wall -I src -c src/IntVector.c -o src/IntVector.o
11
12 clean:
13     rm -rf src/*.o src/main
```