

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. ИС-242

«28» марта 2023 г.

/Пеалкиви Д.Я./

Проверил:

Ст. преподаватель Кафедры ВС

«__» апреля 2023 г.

/Фульман В.О./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ЗАДАНИЕ.....	3
ТРЕБОВАНИЯ К РАБОТЕ.....	3
ВЫПОЛНЕНИЕ РАБОТЫ.....	4
ПРИЛОЖЕНИЕ.....	11

ЗАДАНИЕ

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

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

Для начала создаём заголовочный файл “IntVector.h”, подключаем библиотеку <stdlib.h>, для того, чтобы обращаться к этому заголовочному файлу в других файлах, создаем структуру, которая будет содержать в себе 3 переменные, которые в дальнейшем будут хранить информацию, связанную с массивом, а также записываем все данные функции и то, что они принимают на вход:

IntVector.h

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

typedef struct
{
    int *data;
    int size;
    int capacity;
} IntVector;

IntVector *int_vector_new(size_t initial_capacity);
IntVector *int_vector_copy(const IntVector *v);
void int_vector_free(IntVector *v);
int int_vector_get_item(const IntVector *v, size_t index);
size_t int_vector_get_size(const IntVector *v);
int int_vector_push_back(IntVector *v, int item);
void int_vector_set_item(IntVector *v, size_t index, int item);
size_t int_vector_get_capacity(const IntVector *v);
void int_vector_pop_back(IntVector *v);
int int_vector_shrink_to_fit(IntVector *v);
int int_vector_resize(IntVector *v, size_t new_size);
int int_vector_reserve(IntVector *v, size_t new_capacity);
```

Далее создаем файл “IntVector.c”, в котором будет расписана работа функций, перечисленных в заголовочном файле “IntVector.h”:

Подключаем библиотеку <stdlib.h> и наш заголовочный файл “IntVector.h”:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "IntVector.h"
```

Функция `int_vector_new`:

```
IntVector *int_vector_new(size_t initial_capacity)
{
    IntVector *t = malloc(sizeof(IntVector));
    if (t == NULL)
    {
        return NULL;
    }

    t -> data = malloc(initial_capacity * sizeof(int));
    if(t -> data == NULL)
    {
        free(t);
        return NULL;
    }
    t -> size = 0;
    t -> capacity = initial_capacity;
    return t;
}
```

В этой функции в переменную `t` записывается размер всей структуры, но если в структуре не будет переменной, то нам вернется значение `NULL`. После идёт выделение памяти для переменной `data` и запись в переменную `capacity` числа, которое было у нас на входе.

Функция `int_vector_copy`:

```
IntVector *int_vector_copy(const IntVector *v)
{
    IntVector *t=malloc(sizeof(IntVector));
    if (t == NULL)
        return NULL;
    t->data=malloc(v->capacity*sizeof(int));
    if (t->data==NULL)
    {
        free(t);
        return NULL;
    }
    memcpy(t->data, v->data, sizeof(int) * v->capacity);
    t->size=v->size;
    t->capacity=v->capacity;
    return t;
}
```

В этой функции мы копируем созданный массив в другой, при условии, что изначальный массив не равен значению NULL.

Функция `int_vector_push_back`:

```
int int_vector_push_back(IntVector *v, int item)
{
    if (v->size < v->capacity){
        v->data[v->size] = item;
        v->size++;
    }
    else {
        v->capacity *= 2;

        int *t = realloc(v->data, v->capacity * sizeof(int));
        if (t == NULL)
            return -1;
        v->data = t;
        v->data[v->size] = item;
        v->size++;
    }
    return 0;
}
```

В этой функции проверяем, хватает ли нам места вписать входное число в конец массива. Если места не хватает, то переменную `capacity` увеличиваем в два раза, записывая новый размер в переменную `t` и уже записываем входное число.

Функция `int_vector_shrink_to_fit`:

```
int int_vector_shrink_to_fit(IntVector *v)
{
    if (v->size < v->capacity)
    {
        v->capacity = v->size;
        int *t = realloc(v->data, v->capacity * sizeof(int));
        if (t == NULL)
        {
            return -1;
        }
        v->data = t;
        return 0;
    }
    return -1;
}
```

В этой функции сравниваем размеры переменных `size` и `capacity`. Переменная `size` должна быть меньше переменной `capacity`, и если это не так, то возвращаем `-1` как ошибку, а если условие верно, то переменная `capacity` будет равна переменной `size`. Также, в переменную `t` записываем новый размер для переменной `data` и после изменяем переменную `data`.

Функция `int_vector_resize`:

```
int int_vector_resize(IntVector *v, size_t new_size)
{
    if ((new_size > v->size) && (v->capacity > new_size))
    {
        for (int i = v->size; i < new_size; i++)
        {
            v->data[i] = 0;
        }
        v->size = new_size;
    }
    if (v->size == new_size)
    {
        return 0;
    }
    if (new_size < v->size)
    {
        return -1;
    }
    return 0;
}
```

В этой функции получаем на вход переменную `new_size` и сравниваем её с переменной `size` и переменной `capacity`, эта разница между `new_size` и `size` должна быть заполнена нулями, а в случае если переменная `size` больше, чем переменная `new_size`, то возвращаем `-1`, то есть ошибку.

Функция `int_vector_reserve`:

```
int int_vector_reserve(IntVector *v, size_t new_capacity)
{
    if (new_capacity > v->capacity)
    {
        int *z = realloc(v->data, new_capacity * sizeof(int));
        if (!z)
        {
            return -1;
        }
        v->capacity = new_capacity;
        v->data = z;
        return 0;
    }
    else
    {
        return -1;
    }
}
```

В этой функции значение переменной `new_capacity` должно быть больше значения переменной `capacity`. Если условие верно, то вместе с переменной `capacity` меняется переменная `data`. В переменную `z` вписываем новое значение для переменной `data` и изменяем переменную `capacity` и переменную `data`.

ПРИЛОЖЕНИЕ

IntVector.h

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

typedef struct
{
    int *data;
    int size;
    int capacity;
} IntVector;

IntVector *int_vector_new(size_t initial_capacity);
IntVector *int_vector_copy(const IntVector *v);
void int_vector_free(IntVector *v);
int int_vector_get_item(const IntVector *v, size_t index);
size_t int_vector_get_size(const IntVector *v);
int int_vector_push_back(IntVector *v, int item);
void int_vector_set_item(IntVector *v, size_t index, int item);
size_t int_vector_get_capacity(const IntVector *v);
void int_vector_pop_back(IntVector *v);
int int_vector_shrink_to_fit(IntVector *v);
int int_vector_resize(IntVector *v, size_t new_size);
int int_vector_reserve(IntVector *v, size_t new_capacity);
```

IntVector.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "IntVector.h"

IntVector *int_vector_new(size_t initial_capacity)
{
    IntVector *t = malloc(sizeof(IntVector));
    if (t == NULL)
    {
        return NULL;
    }

    t->data = malloc(initial_capacity * sizeof(int));
    if (t->data == NULL)
    {
        free(t);
        return NULL;
    }
    t->size = 0;
    t->capacity = initial_capacity;
    return t;
}
```

```

IntVector *int_vector_copy(const IntVector *v)
{
    IntVector *t=malloc(sizeof(IntVector));
    if (t == NULL)
        return NULL;
    t->data=malloc(v->capacity*sizeof(int));
    if (t->data==NULL)
    {
        free(t);
        return NULL;
    }
    memcpy(t->data, v->data, sizeof(int) * v->capacity);
    t->size=v->size;
    t->capacity=v->capacity;
    return t;
}

void int_vector_free(IntVector *v)
{
    free(v->data);
    free(v);
}

int int_vector_get_item(const IntVector *v, size_t index)
{
    return v->data[index];
}

void int_vector_set_item(IntVector *v, size_t index, int item)
{
    if (index<=v->capacity)
        v->data[index]=item;
    v->size++;
}

size_t int_vector_get_size(const IntVector *v)
{
    return v->size;
}

size_t int_vector_get_capacity(const IntVector *v)
{
    return v->capacity;
}

```

```

int int_vector_push_back(IntVector *v, int item)
{
    if (v->size < v->capacity){
        v->data[v->size] = item;
        v->size++;
    }
    else {
        v->capacity *= 2;

        int *t = realloc(v->data, v->capacity * sizeof(int));
        if (t == NULL)
            return -1;
        v->data = t;
        v->data[v->size] = item;
        v->size++;
    }
    return 0;
}

void int_vector_pop_back(IntVector *v)
{
    if (v -> size > 0)
        v -> size--;
}

int int_vector_shrink_to_fit(IntVector *v)
{
    if (v->size < v->capacity)
    {
        v->capacity = v->size;
        int *t = realloc(v->data, v->capacity * sizeof(int));
        if (t == NULL)
        {
            return -1;
        }
        v->data = t;
        return 0;
    }
    return -1;
}

```

```

int int_vector_resize(IntVector *v, size_t new_size)
{
    if ((new_size > v->size) && (v->capacity > new_size))
    {
        for (int i = v->size; i < new_size; i++)
        {
            v->data[i] = 0;
        }
        v->size = new_size;
    }
    if (v->size == new_size)
    {
        return 0;
    }
    if (new_size < v->size)
    {
        return -1;
    }
    return 0;
}

int int_vector_reserve(IntVector *v, size_t new_capacity)
{
    if (new_capacity > v->capacity)
    {
        int *z = realloc(v->data, new_capacity * sizeof(int));
        if (!z)
        {
            return -1;
        }
        v->capacity = new_capacity;
        v->data = z;
        return 0;
    }
    else
    {
        return -1;
    }
}

void print_vector(IntVector *v)
{
    for(int i = 0; i < v->size; i++)
        printf("%d ", v->data[i]);
    printf("\n");
    printf("IntVector  \n data = %p\n size = %d\n capacity = %d\n", v->data, v->size,
v->capacity);
    printf("\n");
}

```

```

main.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "IntVector.h"

int main()
{
    printf("Введите capacity:");
    int x; scanf("%d",&x);
    while (x<=0)
    {
        printf("Error");
    }
    IntVector *array = int_vector_new(x);
    print_vector(array);
    for (int i = 0; i < array->capacity; i++)
    {
        int_vector_push_back(array,i);
        printf("%d\n",array -> data[i]);
    }
    print_vector(array);
    IntVector *a = int_vector_copy(array);
    print_vector(array);
    printf("vector a\n");
    printf("%p\n",a);
    for (int i = 0; i < array->capacity; i++) {
        printf("%d\n", a->data[i]);
    }
    printf("\n");
    printf("Get Item\n");
    int z;
    printf("Из какой ячейки взять значение?\n");
    scanf("%d",&x);
    while (x>array->capacity)
    {
        printf("Error");
    }
    z = int_vector_get_item(array,x);
    printf("%d\n",z);
    print_vector(array);
    printf("set item\n");
    printf("Какую ячейку изменить?\n");
    int y; scanf("%d",&y);
    printf("На что изменить?\n");
    scanf("%d",&x);
    while (y>array->capacity)
    {
        printf("Error");
    }
}

```

```

int_vector_set_item(array,y,x);
for (int i = 0; i < array->capacity; i++) {
    printf("%d\n", array->data[i]);
}
size_t s = int_vector_get_size(array);
size_t jos = int_vector_get_capacity(array);
printf("size array = %ld\n",s);
printf("capacity array = %ld\n",jos);
printf("\n");
print_vector(array);
printf("push back\n");
    printf("Какую цифру добавить в конец?\n");
    scanf("%d",&x);
int_vector_push_back(array,x);
print_vector(array);
printf("\n");
printf("pop back\n");
int_vector_pop_back(array);
print_vector(array);
printf("shrink to fit\n");
int_vector_shrink_to_fit(array);
print_vector(array);
    printf("Новое значение capacity?\n");
    scanf("%d",&x);
printf("reserve\n");
int_vector_reserve(array,x);
print_vector(array);
    printf("Новое значение size?\n");
    scanf("%d",&x);
printf("resize\n");
int_vector_resize(array,x);
print_vector(array);
int_vector_free(array);
int_vector_free(a);
return 0;
}

```