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-122
«23» февраля 2022 г. _____ Клепче Г.В..

Проверил:
старший преподаватель
Кафедры ВС
«__» февраля 2022 г. _____ Фульман В.О.

Оценка «_____»

Новосибирск 2022

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Реализовать тип данных «Динамический массив целых чисел» — IntVector и основные функции для работы с ним:

```
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)
void int_vector_set_item(IntVector *v, size_t index, int item)
size_t int_vector_get_size(const IntVector *v)
size_t int_vector_get_capacity(const IntVector *v)
int int_vector_push_back(IntVector *v, int item)
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)
```

Разработать тестовое приложение для демонстрации реализованных функций.

ВЫПОЛНЕНИЕ РАБОТЫ

IntVector *int_vector_new(size_t initial_capacity)
Функция создает массив нулевого размера.
В аргументах указывается емкость массива, а результатом функции является указатель на “IntVector”.
В начале с помощью функции “malloc()” выделяется память для структуры ”IntVector”, с помощью ”calloc()” выделяется память для массива необходимой длины.

IntVector *int_vector_copy(const IntVector *v)
Функция создает копию структуры “IntVector”.
С помощью функции “memcpy”, которая принимает на вход указатель на область памяти, куда необходимо скопировать данные, указатель на начало, а так же колличество байт.
Возвращает указатель на копию вектора “v”, если не удалось выделить память “Null”.

void int_vector_free(IntVector *v)
Освобождаем выделенную память с помощью функции “free( )”, принимающей на вход указатель на область памяти, где была выделена память.
Освобождаем память для структуры и массива.

int int_vector_get_item(const IntVector *v, size_t index)
Функция возвращает элемент массива с индексом “index”.

void int_vector_set_item(IntVector *v, size_t index, int item)
Присваивает элементу под номером “index” значение “item”.

size_t int_vector_get_size(const IntVector *v)
Функция принимает на вход указатель на вектор, и возвращает размер массива.

size_t int_vector_get_capacity(const IntVector *v)
Функция принимает на вход указатель на вектор, и возвращает емкость вектора.

int int_vector_push_back(IntVector *v, int item)
Функция помещает в конец массива, значение которое было передано и увеличивает размерность массива на 1.
Если массив уже заполнен его размерность увеличивается в 2 раза с помощью функции “int_vector_reserve()”.

void int_vector_pop_back(IntVector *v)
Функция удаляет последний элемент массива и уменьшает размер массива на 1.

int int_vector_shrink_to_fit(IntVector *v)
Функция уменьшает емкость вектора, до емкости массива.

int int_vector_resize(IntVector *v, size_t new_size)
Функция изменяет размер массива.
Если новый размер массива больше исходного, то добавленные элементы заполняются нулями.
Если новый размер массива меньше исходного, то перевыделение памяти не происходит.

int int_vector_reserve(IntVector *v, size_t new_capacity)
Функция изменяет емкость массива.
Если новая функция больше старой, то перевыделяем память под нужную емкость. Если новый объем меньше либо равен исходному, то ничего не происходит.

ПРИЛОЖЕНИЕ

Структура:

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

IntVector.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "LibIntVector.h"

IntVector *int_vector_new(size_t initial_capacity)
{
    IntVector *vector = malloc(sizeof(IntVector));
    if(vector == NULL){
        return NULL;
    }
    vector->ptr = calloc(initial_capacity, sizeof(int));
    if(vector->ptr == NULL){
        free(vector->ptr);
        return NULL;
    }
    vector->size = 0;
    vector->capacity = initial_capacity;

    return vector;
}

IntVector *int_vector_copy(const IntVector *v)
{
```

```
    IntVector *copy = int_vector_new(v->capacity);
    if(copy->ptr == NULL){
        free(copy->ptr);
        return NULL;
    }

    copy->size = v->size;
    memcpy(copy->ptr, v->ptr, v->size * sizeof(int));
    return copy;
}

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

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

void int_vector_set_item(IntVector *v, size_t index, int item)
{
        v->ptr[index] = item;
}

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->ptr[v->size] = item;
                v->size++;

        } else {
                int t = int_vector_reserve(v, v->capacity * 2);
                if (t == -1){
                        return -1;
       }
                v->ptr[v->size] = item;
                v->size++;

        }

        return 0;

}

void int_vector_pop_back(IntVector *v)
{
   if(v->size != 0){
   v->ptr[v->size - 1] = 0;
   v->size --;
   }
}

int int_vector_shrink_to_fit(IntVector *v)
```

```
{
   if(int_vector_reserve(v, v->size) == 0){
      v->capacity = v->size;
   } else {
      return -1;
   }

        return 0;

}

int int_vector_resize(IntVector *v, size_t new_size)
{
   if (new_size <= v->size){
                int_vector_shrink_to_fit(v);
   }
        if (new_size > v->size) {
                if(new_size > v->capacity){
         int a = int_vector_reserve(v, v->capacity * 2);
         if(a == -1){
            return -1;
         }
      }

                for(int i = new_size; i - (v->size - 1) <= new_size; i++){
         v->ptr[i] = 0;
      }
        }
   v->size = new_size;
        return 0;
}

int int_vector_reserve(IntVector *v, size_t new_capacity)
```

```
{
    if (new_capacity > v->capacity){
                int *t = realloc(v->ptr, new_capacity * sizeof(int));

                if (t == NULL){
                        return -1;
        } else {
                    v->ptr = t;
            v->capacity = new_capacity;
        }
    }
    return 0;
}
```

main.c:

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

int main()
{
    printf("vector_new & copy\n");
    IntVector *v = int_vector_new(5);
    IntVector *copy = int_vector_copy(v);
    printf("v vector: ptr = %p, size = %d, capacity = %d\n", v->ptr, v->size,
v->capacity);
    printf("Copied vector: ptr = %p, size = %d, capacity = %d\n", copy->ptr,
copy->size, copy->capacity);

    printf("\nget_item\n");
    printf("Enter the index you want to get: ");
    int index;
    scanf("%d", &index);
    printf("%d\n", int_vector_get_item(v, index));
```

```
printf("\nset_item\n");
int item;
printf("Enter the index you want to change: ");
scanf("%d", &index);
printf("Enter the number you want to put in index: ");
scanf("%d", &item);
printf("Num in the index before: %d\n", v->ptr[index]);
int_vector_set_item(v, index, item);
printf("Num in the index after: %d\n", v->ptr[index]);
for(int i = 0; i < v->capacity; i++){
    printf("%d ", v->ptr[i]);
}
printf("\n");

printf("\n get_size \n");
printf("Size of the vector v = %ld\n", int_vector_get_size(v));

printf("\n get_capacity \n");
printf("Capacity of the vector v = %ld\n", int_vector_get_capacity(v));

printf("\n push_back \n");
IntVector *a = int_vector_new(10);
for(int k = 0; k < a->capacity; k++){
    a->ptr[k] = rand() % 1001;
    a->size++;
}
item = 0;
printf("Enter the number you want to put in end of mas: ");
scanf("%d", &item);
printf("Before: ");
for(int i = 0; i < a->capacity; i++){
    printf("%d ", a->ptr[i]);
}
printf("\n");
int_vector_push_back(a, item);
printf("After: ");
for(int i = 0; i < a->capacity; i++){
    printf("%d ", a->ptr[i]);
}
printf("capacity = %d, size = %d\n", a->capacity, a->size);
```

```
printf("\n pop_back \n");
printf("The vector before pop_back: ");
for(int i = 0; i < a->capacity; i++){
    printf("%d ", a->ptr[i]);
}
printf("\nsize = %d, capacity = %d\n", a->size, a->capacity);
int_vector_pop_back(a);
printf("After pop_back: ");
for(int i = 0; i < a->capacity; i++){
    printf("%d ", a->ptr[i]);
}
printf("\nsize = %d, capacity = %d\n", a->size, a->capacity);

printf("\n resize \n");
int size;
printf("The size before resize = %d\n", a->size);
printf("Enter the number of the new size: ");
scanf("%d", &size);
int_vector_resize(a,size);
printf("The size after resize = %d\n", a->size);
for(int i = 0; i < a->size; i++){
    printf("%d ", a->ptr[i]);
}
printf("\n");

printf("\n shrink_to_fit \n");
printf("The capacity of the vector before shrink_to fit = %d\n", a->capacity);
if((int_vector_shrink_to_fit(a)) != -1){
printf("The capacity of the vector after shrink_to fit = %d\n", a->capacity);
}

printf("\n reserve \n");
int new_capacity;
printf("The capacity of the vector a before reserve = %d\n", a->capacity);
printf("Enter the new size of capacity: ");
scanf("%d", &new_capacity);
if((int_vector_reserve(a, new_capacity)) != -1){
    printf("The capacity of the vector a after reserve = %d\n", a->capacity);
}
```

```
    printf("Free up memory\n");
    int_vector_free(a);
    int_vector_free(v);
    int_vector_free(copy);
    return 0;
}
```

LIntVector.h:

```
typedef struct{
    int *ptr;
    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);
void int_vector_set_item(IntVector *v, size_t index, int item);
size_t int_vector_get_size(const IntVector *v);
size_t int_vector_get_capacity(const IntVector *v);
int int_vector_push_back(IntVector *v, int item);
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);
```

Makefile:

```
all: main
main:
	gcc src/main.c src/IntVector.c -o main -Wall
clean:
	rm main
run:
	./main
```