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-121  «8» марта 2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Киреев И.А./ |
|  |  |  |
| Проверил:  ст. преп. Кафедры ВС  «9» марта 2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Фульман В.О./ |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2022

**ОГЛАВЛЕНИЕ**

[**ЗАДАНИЕ** 3](#_Toc35593781)

[**ВЫПОЛНЕНИЕ РАБОТЫ** 4](#_Toc35593782)

[**ПРИЛОЖЕНИЕ** 5](#_Toc35593783)

# **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

*Функции:*

*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)*

**ВЫПОЛНЕНИЕ РАБОТЫ**

Добавляю все прототипы функций и структура IntVector в созданный заголовочный файл

IntVector.h

Содержание IntVector.h

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | typedef struct {  int\* pointer;  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);  int 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); |

IntVector.c будет содержать в себе все функции обработки массива

main.c будет содержать в себе код, проверяющий корректность работы функций, описанных в IntVector.c

**Функция N1**

int\_vector\_new(size\_t initial\_capacity) — создает массив нулевого размера с ёмкостью initial\_capacity.

Сначала выделяется память для структуры IntVector, а затем, для массива внутри самой структуры. Если память не выделится для струкуры, либо для массива, то функция вернёт NULL. Функция malloc() возвращает указатель на первый байт выделенной памяти, в функции, в pointer сохраняется указатель на первый байт памяти, выделенной для массива. В случае, если память выделилась корректно, функция возвращает указатель на экземпляр структуры IntVector.

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | IntVector \*int\_vector\_new(size\_t initial\_capacity){  IntVector \*newvector = malloc(sizeof(IntVector));  if (newvector == NULL) return NULL;  newvector->pointer = malloc(initial\_capacity \* sizeof(int));  if (newvector->pointer == NULL){  free(newvector);  return NULL;  }  newvector->size = 0;  newvector->capacity = initial\_capacity;  return newvector;  } |

**Функция N2**

int\_vector\_copy — создает копию вектора типа IntVector.

Если не удается выделить память для структуры или для массива, то функция возвращает NULL. В случае успеха, создается вектор с таким же размером и вместимостью, как и вектор v.

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | IntVector \*int\_vector\_copy(const IntVector \*v){  IntVector \*copyV = malloc(sizeof(IntVector));  if (copyV == NULL){  return NULL;  }  copyV->pointer = malloc(v->capacity \* sizeof(IntVector));  if (copyV == NULL){  free(copyV);  return NULL;  }  for (int x=0; x<v->capacity; x++) copyV->pointer[x] = v->pointer[x];  copyV->capacity = v->capacity;  copyV->size = v->size;  return copyV;  } |

**Функция N3**

int\_vector\_free(IntVector\* v) - освобождает память, выделенную для вектора v.

Функция освобождает память, которая выделялась для вектора v, используя функцию free().

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3 | void int\_vector\_free(IntVector \*v){  free(v);  } |

**Функция N4**

int\_vector\_get\_item(const IntVector \*v, size\_t index) — функция возвращает элемент, хранящийся в массиве вектора v по индексу index

В случае, если индекс, по которому требуется найти элемент, больше, чем размерность массива, функция вернёт -1, в обратном случае — вернёт элемент.

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3  4 | int int\_vector\_get\_item(const IntVector \*v, size\_t index){  if (index > v->size) return -1;  return v->pointer[index];  } |

**Функция N5**

int\_vector\_set\_item(IntVector \*v, size\_t index, int item) — функция присваевает элементу по индексу index массива векторa v, значение item.

В случае, если индекс, по которому надо присвоить элемент, меньше или равен ёмкости, по индексу index присваивается значение item и увеличивается размер. В обратном случае — функция возвращает -1;

Код функции представлен ниже

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int int\_vector\_set\_item(IntVector \*v, size\_t index, int item){  if (index <= v->capacity){  v->pointer[index] = item;  v->size++;  return 0;  }  else return -1;  } |

**Функция N6**

int\_vector\_get\_size(const IntVector \*v) — функция возвращает размер вектора v.

Код функции представлен ниже

|  |  |
| --- | --- |
| 1  2  3 | size\_t int\_vector\_get\_size(const IntVector \*v){  return v->size;  } |

**Функция N7**

int\_vector\_get\_capacity(const IntVector \*v) — функция возвращает ёмкость вектора v.

Код функции представлен ниже

|  |  |
| --- | --- |
| 1  2  3 | size\_t int\_vector\_get\_capacity(const IntVector \*v){  return v->capacity;  } |

**Функция N8**

int\_vector\_push\_back(IntVector \*v, int item) – функция добавляет элемент в конец массива. В случае, если пользователь пытается записать элемент в заполненный массив, емкость массива увеличивается в 2 раза с помощью функции realloc(). Возвращаемое значение: 0, в случае успеха, -1, в случае ошибки.

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  1113  14 | int int\_vector\_push\_back(IntVector \*v, int item){  if (v->size < v->capacity){  v->pointer[v->size] = item;  v->size++;  }  else {  v->capacity\*=2;  int \*p = realloc(v->pointer, v->capacity\*sizeof(int));  if (p == NULL) return -1;  v->pointer = p;  v->pointer[v->size] = item;  v->size++;  }  return 0;  } |

**Функция N9**

int\_vector\_pop\_back(IntVector \*v) – удаляет последний элемент из массива. В случае, если размер массива равен нулю, не возвращает ничего.

|  |  |
| --- | --- |
| 1  2  3 | void int\_vector\_pop\_back(IntVector \*v){  if (v->size != 0) v->size--;  } |

**Функция N10**

int\_vector\_shrink\_to\_fit(IntVector \*v) – функция уменьшает ёмкость массива до размера массива. Возвращает -1, в случае ошибки, 0, в случае успеха. Уменьшение ёмкости массива реализовано с помощью функции realloc().

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int int\_vector\_shrink\_to\_fit(IntVector \*v){  if (v->size > v->capacity) return -1;  v->capacity = v->size;  int \*p = realloc(v->pointer, v->size\*sizeof(int));  if (p == NULL) return -1;  v->pointer = p;  return 0;  } |

**Функция N11**

int\_vector\_resize(IntVector \*v, size\_t new\_size) – функция изменяет размер массива. В случае, если новый размер массива больше исходного, добавленные ячейки заполняются нулями. Если новый размер меньше исходного, отработает функция int\_vector\_shrink\_to\_fit(). В случае корректной работы, функция возвращает 0, иначе, -1.

Код функции представлен ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  1113  14  15  16  17 | int int\_vector\_resize(IntVector \*v, size\_t new\_size){  if (v->size < v->capacity){  if (v->size < new\_size){  int \*p = realloc(v->pointer, sizeof(int));  if (p == NULL) return -1;  v->pointer = p;  for (int x=v->size+1; x<new\_size; x++) v->pointer[x] = 0;  }  else if (v->size == new\_size) return 0;  else {  int chck = int\_vector\_shrink\_to\_fit(v);  if (chck != 0) return -1;  }  v->size = new\_size;  return 0;  }  else return -1;  } |

**Функция N12**

int\_vector\_reserve(IntVector \*v, size\_t new\_capacity) – изменяет ёмкость массива в случае, если новая ёмкость больше исходной. В обратном случае эффекта нет. Ёмкость изменяется с помощью функции realloc().

Код функции представлен ниже

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | int int\_vector\_reserve(IntVector \*v, size\_t new\_capacity){  if (v->capacity < new\_capacity){  int \*p = realloc(v->pointer, new\_capacity\*sizeof(int));  if (p == NULL) return -1;  v->pointer = p;  v->capacity = new\_capacity;  return 0;  }  } |

# **ПРИЛОЖЕНИЕ**

Содержание main.c

**main.c**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67 | #include <stdio.h>  #include <stdlib.h>  #include "IntVector.h"  int main(){  IntVector \*v = int\_vector\_new(5);  IntVector \*cV = int\_vector\_copy(v);  int\_vector\_free(v);  for (int x=0; x<cV->capacity; x++) int\_vector\_set\_item(cV, x, x);  int size = int\_vector\_get\_size(cV);  printf("---BASIC ARRAY---\n");  for (int i=0; i<size; i++){  int item = int\_vector\_get\_item(cV, i);  printf("%d ", item);  }  printf("\n");  printf("\n");  int\_vector\_push\_back(cV, 5);  printf("---ARRAY AFTER PUSHBACK---\n");  for (int j=0; j<cV->size; j++){  int item = int\_vector\_get\_item(cV, j);  printf("%d ", item);  }  printf("\n");  int capacity = int\_vector\_get\_capacity(cV);  printf("Capacity - %d\n", capacity);  size = int\_vector\_get\_size(cV);  printf("Size - %d\n", size);  printf("\n");  int\_vector\_pop\_back(cV);  printf("---ARRAY AFTER POPBACK---\n");  for (int j=0; j<cV->size; j++){  int item = int\_vector\_get\_item(cV, j);  printf("%d ", item);  }  printf("\n");  int\_vector\_resize(cV, 8);  size = int\_vector\_get\_size(cV);  capacity = int\_vector\_get\_capacity(cV);  printf("\n");  printf("---ARRAY AFTER RESIZE---\n");  for (int j=0; j<cV->size; j++){  int item = int\_vector\_get\_item(cV, j);  printf("%d ", item);  }  printf("\n");  printf("size - %d\n", size);  printf("capacity - %d\n", capacity);  int\_vector\_shrink\_to\_fit(cV);  capacity = int\_vector\_get\_capacity(cV);  printf("\n");  printf("---CAPACITY AFTER SHRINKTOFIT---\ncapacity - %d\n", capacity);  int\_vector\_reserve(cV, 10);  capacity = int\_vector\_get\_capacity(cV);  printf("\n");  printf("---CAPACITY AFTER RESERVE---\ncapacity - %d\n", capacity);  return 0;  } |

Данный код работает с функциями, описанными в IntVector.c

6 — создается вектор v, c ёмкостью 5, с помощью функции int\_vector\_new().

8 — оздается копия вектора v-cV, с помощью функции int\_vector\_copy().

10 — освобождается память, выделенная для вектора v.

\*Далее, будет обрабатываться только вектор cV.

12 — в цикле for записываем по индексам в диапазоне от 0 до 4 зачения индексов соответственно с помощью функции int\_vector\_set\_item().

14 — записываем в переменную size результат работы фукции int\_vector\_get\_size() - размер массива.

16-22 — выводятся элементы массива с помощью функции int\_vector\_get\_item().

24 — с помощью функции int\_vector\_push\_back(), добавляется элемент со значением 5.

25-35 — вывод массива после отработки функции int\_vector\_push\_back().

37 — с помощью функции int\_vector\_push\_back(), удаляется последний элемент массива.

38-43 — вывод массива после отработки функции int\_vector\_pop\_back().

45 — изменяется размер массива с помощью функции int\_vector\_resize().

48-56 — вывод массива после отработки функции int\_vector\_resize().

58 — уменьшается ёмкость массива до его размера с помощью функции int\_vector\_shrink\_to\_fit().

60-61 — вывод ёмкости после отработки функции int\_vector\_shrink\_to\_fit().

63 — изменение ёмкости массива до 10 с помощью функции int\_vector\_reserve().