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-221  15 марта 2023 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | Гордов Р.С. |
|  |  |  |
| Проверил:  Старший преподаватель кафедры ВС  «20» марта 2023 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | Фульман В.О. |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[**ЗАДАНИЕ** 3](#_Toc35593781)

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

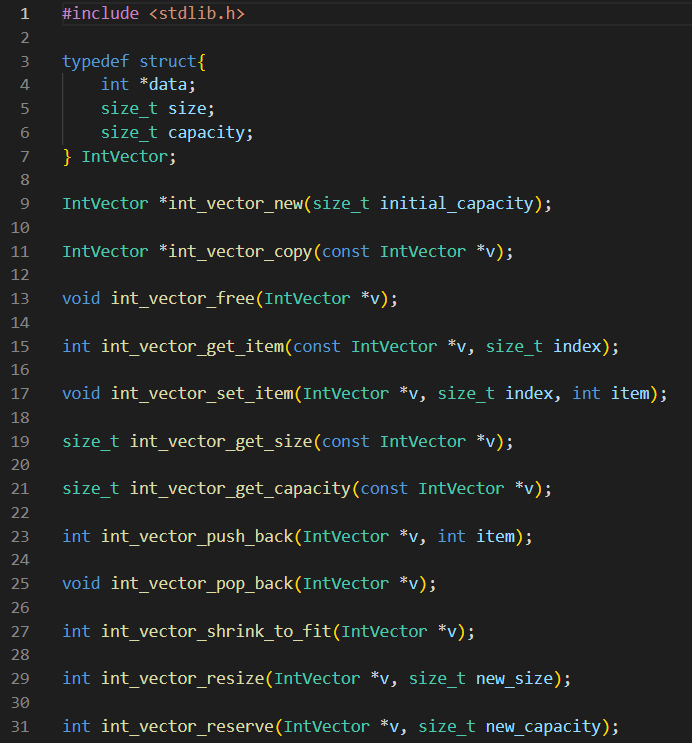
[**ПРИЛОЖЕНИЕ** 5](#_Toc35593783)

# **ЗАДАНИЕ**

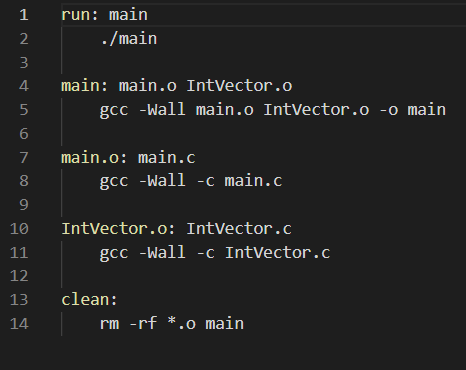
Реализовать тип данных «Динамический массив целых чисел» — IntVector и основные функции для работы с ним. Разработать тестовое приложение для демонстрации реализованных функций.

**ВЫПОЛНЕНИЕ РАБОТЫ**

Создадим файл с расширением .h в нем будет лежать структура, содержащая три поля — данные, размер и емкость нашего массива, а также прототипы функций.



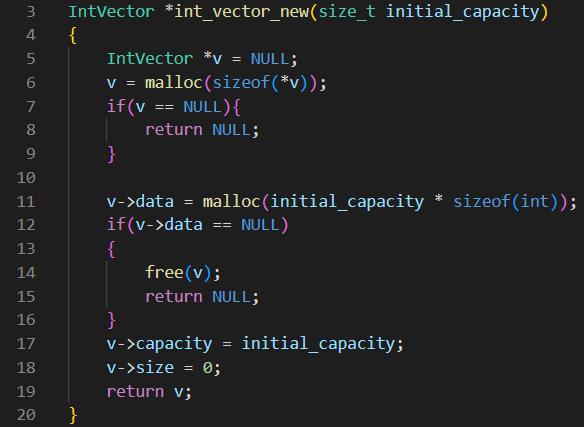
Создадим для удобства запуска программы создадим Makefile.



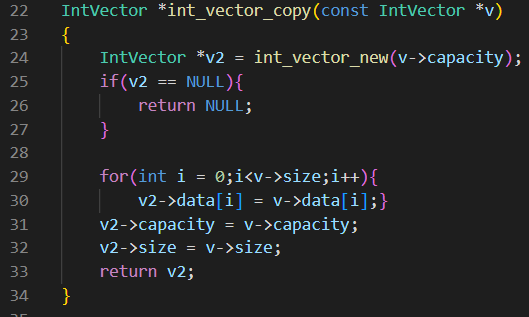
В файле IntVector.c будет реализация функций для работы с нашим вектором.

Напишем функцию, которая создает динамический массив. Она принимает изначальную емкость и возвращает указатель на нашу структуру.

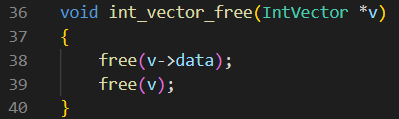
В случае, если не удается выделить место либо под структуру, либо под сами данные, возвращает нулевой указатель, если удалось выделить память, записываем в структуру начальные значение емкости и размера.



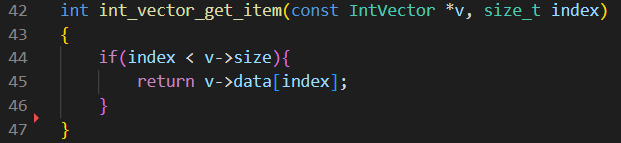
Реализуем функцию, которая возвращает указатель на копию нашего динамического массива. Она создает новый вектор, в который копирует все данные из предыдущего, и возвращает указатель на новый вектор.



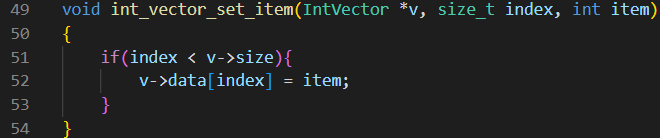
Функция int\_vector\_free принимает указатель на наш массив и освобождает выделенную для него память.



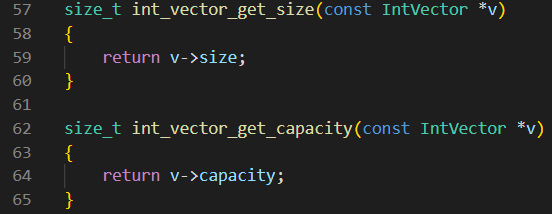
Функция int\_vector\_get\_item возвращает элемент под определенным индексом, если такой индекс существует.



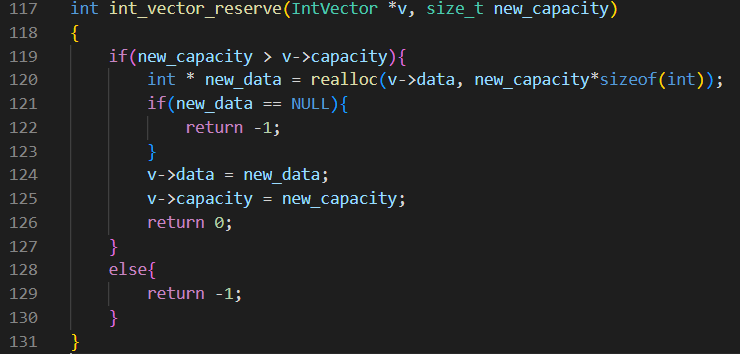
Функция int\_vector\_set\_item изменяет значение под определенным индексом на передаваемое, конечно, если такой индекс существует.



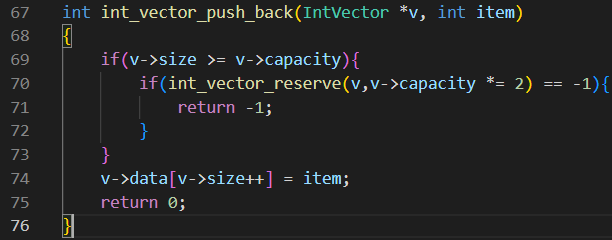
Следующие функции возвращают значения полей size и capacity соответственно.



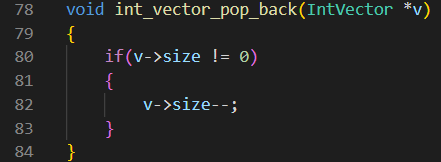
Функция int\_vector\_reserve увеличивает емкость нашего вектора и возвращает 0 в случае успеха. Если память выделить не удалось или новая емкость меньше предыдущей возвращает -1.



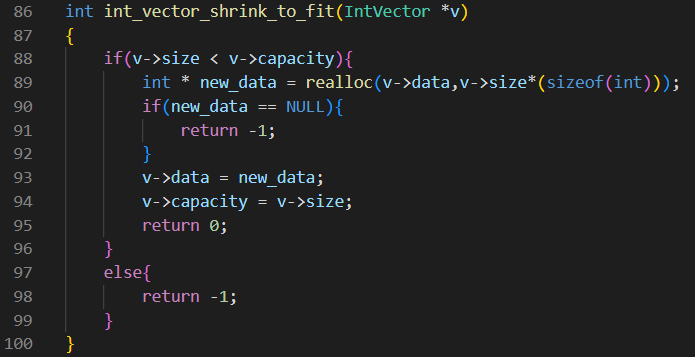
Функция int\_vector\_push\_back, записывает новый элемент в конец массива, в случае если нам не хватает размера, мы увеличиваем емкость в два раза. Возвращает 0 в случае успешной записи, и возвращает -1, если нам не удалось выделить память.



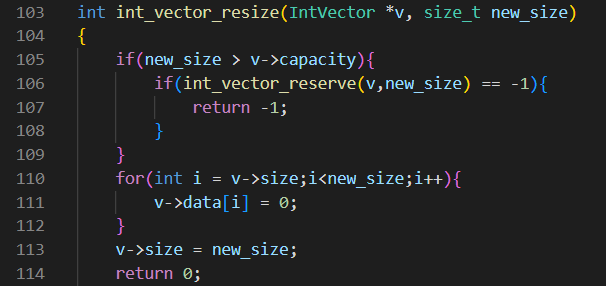
Функция int\_vector\_pop\_back удаляет последний элемент массива.



Функция int\_vector\_shrink\_to\_fit уменьшает емкость массива до его размера и возвращает 0 в случае успеха и -1 в случае ошибки.



Функция int\_vector\_resize изменяет размер массива и новые элементы инициализирует нулями, в случае если новый размер больше емкости, мы увеличиваем емкость в два раза. Возвращает 0 в случае успеха и -1 в случае ошибки.



# **ПРИЛОЖЕНИЕ**

<https://github.com/Herroa/proga2023>

**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 | #include <assert.h>  #include <stddef.h>  #include <stdint.h>  **size\_t** **encode\_varint**(**uint32\_t** value, **uint8\_t**\* buf)  {  assert(buf != NULL);  **uint8\_t**\* cur = buf;  **while** (value >= **0x80**) {  **const** **uint8\_t** byte = (value & **0x7f**) | **0x80**;  \*cur = byte;  value >>= **7**;  ++cur;  }  \*cur = value;  ++cur;  **return** cur - buf;  }  **uint32\_t** **decode\_varint**(**const** **uint8\_t**\*\* bufp)  {  **const** **uint8\_t**\* cur = \*bufp;  **uint8\_t** byte = \*cur++;  **uint32\_t** value = byte & **0x7f**;  **size\_t** shift = **7**;  **while** (byte >= **0x80**) {  byte = \*cur++;  value += (byte & **0x7f**) << shift;  shift += **7**;  }  \*bufp = cur;  **return** value;  } |