МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5b

по дисциплине: Основы программирования
тема: Реализация структуры данных «Вектор»

Выполнил: ст. группы ПВ-211
Чувилко Илья Романович

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Белгород 2021 г.

Цель работы: усовершенствование навыков в создании библиотек, получение навыков работы с системой контроля версий `git`

Содержание отчета:

- Тема лабораторной работы.
- Цель лабораторной работы.
- Ссылка на открытый репозиторий с решением.
- Исходный код файлов:
  - vector.h / vector.c
  - *vectorVoid.h / vectorVoid.c – main.c
- Результат выполнения команд
- Выводы по работе

Требования:

Обратите особое внимание на задания к лабораторной работе. В частности, на требование к коммитам в процессе выполнения работы. Если работа будет выполнена без них или будут отсутствовать промежуточные коммиты, она не будет засчитана. С целью улучшения умения чтения текста лабораторной, будет выдано дополнительное задание.

Ссылка на репозиторий:
https://github.com/ChuvilkoDEV/lab5b/tree/master

Содержимое файла vector.h:

```c
#ifndef LAB_5B_VECTOR_H
#define LAB_5B_VECTOR_H

#include <stdio.h>
#include <malloc.h>
#include <stdbool.h>

typedef struct vector {
  int *data; // указатель на элементы вектора
  size_t size; // размер вектора
  size_t capacity; // вместимость вектора
} vector;

// Возвращает структуру-дескриптор вектор из n значений.
vector createVector(size_t n);

// Выводит на экран вектор и информацию о нем
void outputVector(vector v);

// Изменяет количество памяти, выделенное под хранение элементов вектора.
void reserve(vector *v, size_t newCapacity);

// Удаляет элементы из контейнера, но не освобождает выделенную память.
void clear(vector *v);

// Освобождает память, выделенную под неиспользуемые элементы
void shrinkToFit(vector *v);

// Освобождает память, выделенную вектору
void deleteVector(vector *v);

// Возвращает true, если вектор пустой. Иначе false.
bool isEmpty(vector *v);

// Возвращает true, если вектор полный. Иначе false.
bool isFull(vector *v);

// Возвращает i-ый элемент вектора v. Тело функции – одна строка кода.
int getVectorValue(vector *v, size_t i);
```

```
// Добавляет элемент x в конец вектора v. Если вектор заполнен,
// увеличивает количество выделенной ему памяти в 2 раза, используя reserve.
void pushBack(vector *v, int x);

// Удаляет последний элемент из вектора. Функция ’выкидывает’ в поток
// ошибок сообщение, если вектор пуст и заканчивает выполнение с кодом 1.
void popBack(vector *v);

// Возвращает указатель на index-ый элемент вектора. Если осуществляется
// попытка получить доступ вне пределов используемых элементов вектора,
// в поток ошибок выводится ошибка: "IndexError: a[index] is not exists",
// где в качестве index указывается позиция элемента, к которому пытались
// осуществить доступ.
int* atVector(vector *v, size_t index);

// Возвращает указатель на последний элемент вектора
int* back(vector *v);

// Возвращает указатель на первый элемент вектора
int* front(vector *v);

#endif //LAB_5B_VECTOR_H
```

Содержимое файла vector.c:

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

// Возвращает структуру-дескриптор вектор из n значений.
vector createVector(size_t n) {
  vector v;
  v.data = (int *) malloc(n * sizeof(int));
  if (v.data == 0) {
    fprintf(stderr, "bad alloc ");
    exit(1);
  }
  v.size = 0;
  v.capacity = n;
  return v;
}

void outputVector(vector v) {
  printf("data = (");
  for (int i = 0; i < v.size; i++)
    printf("%d, ", v.data[i]);
  printf("\b\b)\n");
  printf("size = %zu\ncapacity = %zu\n", v.size, v.capacity);
}

// Изменяет количество памяти, выделенное под хранение элементов вектора.
void reserve(vector *v, size_t newCapacity) {
  if (newCapacity == 0) {
    v->data = NULL;
    v->size = 0;
  } else {
    if (newCapacity < v->size)
      v->size = newCapacity;
    v->data = (int *) realloc(v->data, newCapacity * sizeof(int));
    if (v->data == 0) {
      fprintf(stderr, "bad alloc ");
      exit(1);
    }
  }
```

```c
  v->capacity = newCapacity;
}

// Удаляет элементы из контейнера, но не освобождает выделенную память.
void clear(vector *v) {
  v->size = 0;
}

// Освобождает память, выделенную под неиспользуемые элементы
void shrinkToFit(vector *v) {
  v->data = (int *) realloc(v->data, v->size * sizeof(int));
  v->capacity = v->size;
}

// Освобождает память, выделенную вектору
void deleteVector(vector *v) {
  free(v->data);
}

// Возвращает true, если вектор пустой. Иначе false.
bool isEmpty(vector *v) {
  return v->size == 0 ? true : false;
}

// Возвращает true, если вектор полный. Иначе false.
bool isFull(vector *v) {
  return v->size == v->capacity ? true : false;
}

// Возвращает i-ый элемент вектора v. Тело функции – одна строка кода.
int getVectorValue(vector *v, size_t i) {
  return v->data[i];
}

// Добавляет элемент x в конец вектора v. Если вектор заполнен,
// увеличивает количество выделенной ему памяти в 2 раза, используя reserve.
void pushBack(vector *v, int x) {
  if (isFull(v))
    reserve(v, v->capacity == 0 ? 1 : v->capacity * 2);
  v->data[v->size++] = x;
}

// Удаляет последний элемент из вектора. Функция 'выкидывает' в поток
// ошибок сообщение, если вектор пуст и заканчивает выполнение с кодом 1.
void popBack(vector *v) {
  if (isEmpty(v)) {
    fprintf(stderr, "vector is empty");
    exit(1);
  }
  v->size--;
}

// Возвращает указатель на index-ый элемент вектора. Если осуществляется
// попытка получить доступ вне пределов используемых элементов вектора,
// в поток ошибок выводится ошибка: "IndexError: a[index] is not exists",
// где в качестве index указывается позиция элемента, к которому пытались
// осуществить доступ.
int* atVector(vector *v, size_t index) {
  if (index > v->size)
    fprintf(stderr, "a[%zu] is not exist", index);
  return &v->data[index];
}
```

```
// Возвращает указатель на последний элемент вектора
int* back(vector *v) {
  if (isEmpty(v))
    fprintf(stderr, "vector is empty");
  return &v->data[v->size - 1];
}

// Возвращает указатель на первый элемент вектора
int* front(vector *v) {
  if (isEmpty(v))
    fprintf(stderr, "vector is empty");
  return &v->data[0];
}
```

Тесты для Vector.h:

```
void test_createVector_emptyVector() {
  vector v = createVector(0);

  assert(v.size == 0);
  assert(v.capacity == 0);

  deleteVector(&v);
}

void test_createVector_notEmptyVector() {
  vector v = createVector(5);

  assert(v.size == 0);
  assert(v.capacity == 5);

  deleteVector(&v);
}

void test_createVector() {
  test_createVector_emptyVector();
  test_createVector_notEmptyVector();
}

void test_reserve_newCapacityZero() {
  vector v = createVector(5);

  reserve(&v, 0);

  assert(v.data == 0);
  assert(v.size == 0);
  assert(v.capacity == 0);
  deleteVector(&v);
}

void test_reserve_newCapacitySmallerSize() {
  vector v = createVector(5);

  pushBack(&v, 0);
  pushBack(&v, 1);
  pushBack(&v, 2);
  pushBack(&v, 3);
  reserve(&v, 3);

  int resArray[] = {0, 1, 2};

  assert(isEqualArray(v.data, v.size, resArray, 3));
  assert(v.size == 3);
  assert(v.capacity == 3);
  deleteVector(&v);
}
```

```
void test_reserve_newCapacityBiggerSize() {
  vector v = createVector(5);

  pushBack(&v, 0);
  pushBack(&v, 1);
  pushBack(&v, 2);
  pushBack(&v, 3);
  reserve(&v, 8);

  int resArray[] = {0, 1, 2, 3};

  assert(isEqualArray(v.data, v.size, resArray, 4));
  assert(v.size == 4);
  assert(v.capacity == 8);
  deleteVector(&v);
}

void test_reserve() {
  test_reserve_newCapacityZero();
  test_reserve_newCapacitySmallerSize();
  test_reserve_newCapacityBiggerSize();
}

void test_shrinkToFit() {
  vector v = createVector(6);

  pushBack(&v, 0);
  pushBack(&v, 1);
  pushBack(&v, 2);
  pushBack(&v, 3);
  shrinkToFit(&v);

  int resArray[] = {0, 1, 2, 3};

  assert(isEqualArray(v.data, v.size, resArray, 4));
  assert(v.size == 4);
  assert(v.capacity == 4);
  deleteVector(&v);
}

void test_pushBack_emptyVector1() {
  vector v = createVector(2);
  pushBack(&v, 1);

  assert(v.data[0] == 1);
  assert(v.capacity == 2);
  deleteVector(&v);
}

void test_pushBack_emptyVector2() {
  vector v = createVector(0);
  pushBack(&v, 1);

  assert(v.data[0] == 1);
  assert(v.capacity == 1);
  deleteVector(&v);
}

void test_pushBack_emptyVector() {
  test_pushBack_emptyVector1();
  test_pushBack_emptyVector2();
}
```

```
void test_pushBack_fullVector() {
  vector v = createVector(2);
  pushBack(&v, 1);
  pushBack(&v, 2);
  pushBack(&v, 3);
  assert(v.data[2] == 3);

  deleteVector(&v);
}

void test_popBack_notEmptyVector() {
  vector v = createVector(0);
  pushBack(&v, 10);

  assert (v.size == 1);
  popBack(&v);
  assert (v.size == 0);
  assert (v.capacity == 1);

  deleteVector(&v);
}

void test_atVector_notEmptyVector() {
  vector v = createVector(2);
  pushBack(&v, 1);
  pushBack(&v, 2);

  assert(atVector(&v, 0) == &v.data[0]);

  deleteVector(&v);
}

void test_atVector_requestToLastElement() {
  vector v = createVector(2);
  pushBack(&v, 1);
  pushBack(&v, 2);

  assert(atVector(&v, 1) == &v.data[1]);

  deleteVector(&v);
}

void test_back_oneElementInVector() {
  vector v = createVector(3);
  pushBack(&v, 1);
  pushBack(&v, 2);

  assert(back(&v) == &v.data[1]);

  deleteVector(&v);
}

void test_front_oneElementInVector() {
  vector v = createVector(3);
  pushBack(&v, 1);
  pushBack(&v, 2);

  assert(front(&v) == &v.data[0]);

  deleteVector(&v);
}
```

Содержимое файла voidVector.h:

```c
#ifndef LAB_5B_VOIDVECTOR_H
#define LAB_5B_VOIDVECTOR_H

#include <stdbool.h>
# include <limits.h>

typedef struct vectorVoid {
  void *data; // указатель на нулевой элемент вектора

  size_t size; // размер вектора
  size_t capacity; // вместимость вектора
  size_t baseTypeSize; // размер базового типа:
  // например, если вектор хранит int -
  // то поле baseTypeSize = sizeof(int)
  // если вектор хранит float -
  // то поле baseTypeSize = sizeof(float)
} vectorVoid;


// возвращает структуру-дескриптор вектор из n значений.
vectorVoid createVectorV(size_t n, size_t baseTypeSize);

// изменяет количество памяти, выделенное под хранение элементов вектора.
void reserveV(vectorVoid *v, size_t newCapacity);

// Обнуляет размер вектора
void clearV(vectorVoid *v);

// освобождает память, выделенную под неиспользуемые элементы
void shrinkToFitV(vectorVoid *v);

// Освобождает память, выделенную вектору
void deleteVectorV(vectorVoid *v);

// Возвращает true, если вектор пустой. False в ином случае.
bool isEmptyV(vectorVoid *v);

// Возвращает true, если вектор полный. False в ином случае.
bool isFullV(vectorVoid *v);

// записывает по адресу destination index-ый элемент вектора v.
void getVectorValueV(vectorVoid *v, size_t index, void *destination);

// записывает на index-ый элемент вектора v значение, расположенное по
// адресу source
void setVectorValueV(vectorVoid *v, size_t index, void *source);

// Удаляет последний элемент из вектора. Функция 'выкидывает' в поток
// ошибок сообщение, если вектор пуст и заканчивает выполнение с кодом 1.
void popBackV(vectorVoid *v);

// Добавляет элемент x в конец вектора v. Если вектор заполнен,
// увеличивает количество выделенной ему памяти в 2 раза, используя reserve.
void pushBackV(vectorVoid *v, void *source);

#endif //LAB_5B_VOIDVECTOR_H
```

Содержимое файла voidVector.c:

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

// возвращает структуру-дескриптор вектор из n значений.
vectorVoid createVectorV(size_t n, size_t baseTypeSize) {
  vectorVoid v;
  v.data = (void *) malloc(n * baseTypeSize);
  if (v.data == 0) {
    fprintf(stderr, "bad alloc ");
    exit(1);
  }
  v.size = 0;
  v.capacity = n;
  v.baseTypeSize = baseTypeSize;
  return v;
}

// изменяет количество памяти, выделенное под хранение элементов вектора.
void reserveV(vectorVoid *v, size_t newCapacity) {
  if (newCapacity == 0) {
    v->data = NULL;
    v->size = 0;
  } else {
    if (newCapacity < v->size)
      v->size = newCapacity;
    v->data = (void *) realloc(v->data, newCapacity * v->baseTypeSize);
  }
  v->capacity = newCapacity;
}

// Обнуляет размер вектора
void clearV(vectorVoid *v) {
  v->size = 0;
}

// освобождает память, выделенную под неиспользуемые элементы
void shrinkToFitV(vectorVoid *v) {
  v->data = (void *) realloc(v->data, v->size * v->baseTypeSize);
  v->capacity = v->size;
}

// Освобождает память, выделенную вектору
void deleteVectorV(vectorVoid *v) {
  free(v->data);
  v->size = 0;
  v->capacity = 0;
  v->baseTypeSize = 0;
}

// Возвращает true, если вектор пустой. False в ином случае.
bool isEmptyV(vectorVoid *v) {
  return v->size == 0 ? true : false;
}

// Возвращает true, если вектор полный. False в ином случае.
bool isFullV(vectorVoid *v) {
  return v->size == v->capacity ? true : false;
}
```

```c
// записывает по адресу destination index-ый элемент вектора v.
void getVectorValueV(vectorVoid *v, size_t index, void *destination) {
  if (index >= v->capacity)
    fprintf(stderr, "index > capacity :'( ");
  char *source = (char *) v->data + index * v->baseTypeSize;
  memcpy(destination, source, v->baseTypeSize);
}

// записывает на index-ый элемент вектора v значение, расположенное по
// адресу source
void setVectorValueV(vectorVoid *v, size_t index, void *source) {
  if (index >= v->capacity)
    fprintf(stderr, "index > capacity :'( ");
  char *destination = (char *) v->data + index * v->baseTypeSize;
  memcpy(destination, source, v->baseTypeSize);
}

void popBackV(vectorVoid *v) {
  if (isEmptyV(v)) {
    fprintf(stderr, "vector is empty");
    exit(1);
  }
  (v->size)--;
}

void pushBackV(vectorVoid *v, void *source) {
  if (isFullV(v))
    reserveV(v, v->capacity == 0 ? v->capacity + 1 : v->capacity * 2);
  setVectorValueV(v, v->size++, source);
}
```

Тесты для voidVector.h

```c
void test_getVectorValueV() {
  vectorVoid v = createVectorV(5, sizeof(int));
  int source[] = {1, 2, 3, 4, 5};
  for (int i = 0; i < 5; i++)
    setVectorValueV(&v, i, &source[i]);

  assert(isEqualArray(v.data, v.size, source, 5));
  deleteVectorV(&v);
}

void test_setVectorValueV() {
  vectorVoid v = createVectorV(5, sizeof(int));
  int source[] = {1, 2, 3, 4, 5};
  for (int i = 0; i < 5; i++)
    setVectorValueV(&v, i, &source[i]);

  assert(isEqualArray(v.data, v.size, source, 5));
  deleteVectorV(&v);
}

void test_pushBack_fromManual1() {
  size_t n;
  scanf("%zd", &n);

  vectorVoid v = createVectorV(0, sizeof(int));
  for (int i = 0; i < n; i++) {
    int x;
    scanf("%d", &x);

    pushBackV(&v, &x);
  }
```

```c
for (int i = 0; i < n; i++) {
    int x;
    getVectorValueV(&v, i, &x);

    printf("%d ", x);
  }
}

void test_pushBack_fromManual2() {
  size_t n;
  scanf("%zd", &n);

  vectorVoid v = createVectorV(0, sizeof(float));
  for (int i = 0; i < n; i++) {
    float x;
    scanf("%f", &x);

    pushBackV(&v, &x);
  }

  for (int i = 0; i < n; i++) {
    float x;
    getVectorValueV(&v, i, &x);

    printf("%f ", x);
  }
}
```

Коммиты для vector.c:

```
commit 83455f622f14c6ef9ad8c5218ba416f625c3c009
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 10:08:36 2022 +0300

    Specification vector.h

 libs/data_structures/vector/vector.c | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)

commit 041dfed530822ba48b638418f08b728d0e52303e
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 00:32:23 2022 +0300

    Specification

 libs/data_structures/vector/vector.c | 27 ++++++++++++++++++++++++---
 1 file changed, 24 insertions(+), 3 deletions(-)

commit b35665eed7cc98605fae01935ee35892424c18c5
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 20:32:37 2022 +0300
:...skipping...
commit 83455f622f14c6ef9ad8c5218ba416f625c3c009
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 10:08:36 2022 +0300

    Specification vector.h
```

```
commit b35665eed7cc98605fae01935ee35892424c18c5
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 20:32:37 2022 +0300

    append push / pop functions

 libs/data_structures/vector/vector.c | 18 +++++++++---------
 1 file changed, 9 insertions(+), 9 deletions(-)

commit c6d435d865f0378277ee801b88c366a376fd96a2
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 14:02:38 2022 +0300

    refactoring #2

 libs/data_structures/vector/vector.c | 10 +++++-----
 1 file changed, 5 insertions(+), 5 deletions(-)

commit b4d6746d9aebf1b8d55a7b9518f174c192ccdd68
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 12:54:17 2022 +0300

    refactoring #1

 libs/data_structures/vector/vector.c | 7 +++++--
 1 file changed, 5 insertions(+), 2 deletions(-)
```

```
commit 00b296e874323a1830d53a29ff5122a724ee9a46
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 12:27:37 2022 +0300

    append access functions

 libs/data_structures/vector/vector.c | 5 +++++
 1 file changed, 5 insertions(+)

commit 45d49b6fc51121dca63510e0c3c82141caafcb0f
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 00:21:08 2022 +0300

    index vector

 libs/data_structures/vector/vector.c | 19 +++++++++++++++++---
 1 file changed, 16 insertions(+), 3 deletions(-)

commit cbcca423c28ebfab4f1b4f4dffcd91fe75a1a3a2
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sat Feb 5 19:52:36 2022 +0300

    +lib array

 libs/data_structures/vector/vector.c | 26 ++++++++++++++++++++--------
 1 file changed, 18 insertions(+), 8 deletions(-)
```

```
commit 5f3a00a39fec6151a9d2a14ae0a844f91c9f296d
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sat Feb 5 18:14:56 2022 +0300

    №8

 libs/data_structures/vector/vector.c | 20 ++++++++++++++++++---
 1 file changed, 17 insertions(+), 3 deletions(-)

commit 1d180d157c8be5f277f2aa078fc1a07dde892a43
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 2 19:43:55 2022 +0300

    second

 libs/data_structures/vector/vector.c | 19 ++++++++++++++++++-
 1 file changed, 18 insertions(+), 1 deletion(-)

commit b55fc52df732873d7865ab9215596f30c1dedb16
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 2 17:37:01 2022 +0300

    memory usage of vector

 libs/data_structures/vector/vector.c | 35 ++++++++++++++++++++++++++++++++++++----
 1 file changed, 31 insertions(+), 4 deletions(-)
```

Коммиты для voidVector.c:

```
commit ac2c4baab4ea4718fc1f6f407ba2a008411dcfa5 (HEAD -> master, origin/master)
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 10:22:25 2022 +0300

    Specification voidVector.h

 libs/data_structures/vector/voidVector.c | 4 ++++
 1 file changed, 4 insertions(+)

commit 041dfed530822ba48b638418f08b728d0e52303e
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 00:32:23 2022 +0300

    Specification

 libs/data_structures/vector/voidVector.c | 10 ++++++++--
 1 file changed, 8 insertions(+), 2 deletions(-)

:...skipping...
commit ac2c4baab4ea4718fc1f6f407ba2a008411dcfa5 (HEAD -> master, origin/master)
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 10:22:25 2022 +0300

    Specification voidVector.h

 libs/data_structures/vector/voidVector.c | 4 ++++
 1 file changed, 4 insertions(+)
```

```
commit 041dfed530822ba48b638418f08b728d0e52303e
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Wed Feb 9 00:32:23 2022 +0300

    Specification

 libs/data_structures/vector/voidVector.c | 10 +++++++++--
 1 file changed, 8 insertions(+), 2 deletions(-)

commit 1da3c493317c797f06a51c5947bff54ce2f02602
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Tue Feb 8 20:22:40 2022 +0300

    refactoring + tests

 libs/data_structures/vector/voidVector.c | 14 ++++++++++++----
 1 file changed, 10 insertions(+), 4 deletions(-)
```

```
commit b35665eed7cc98605fae01935ee35892424c18c5
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 20:32:37 2022 +0300

    append push / pop functions

 libs/data_structures/vector/voidVector.c | 51 +++++++++++++++++++++++++++++++++++++++++++++++++++++++------
 1 file changed, 42 insertions(+), 9 deletions(-)

commit 929c54b8af9092df3554d83669e8655caf8b245d
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 14:35:14 2022 +0300

    memory usage of vectorVoid

 libs/data_structures/vector/voidVector.c | 42 ++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 42 insertions(+)

commit 7c7cd919e3bb10fa8a91a303efeba4f480673f70
Author: ChuvilkoDEV <ilyasutton@gmail.com>
Date:   Sun Feb 6 14:28:03 2022 +0300

    init vectorVoid
```

Вывод: усовершенствовал навыки создания библиотек, получил навык работы с системой контроля версий `git`