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 №13

По дисциплине: «Основы программирования»

Тема: Реализация структуры данных «Вектор»

Выполнил: студент группы ВТ-231

Борченко Александр Сергеевич

Проверили:

Черников Сергей Викторович

Новожен Никита Викторович

Белгород 2023

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

Содержание работы:

Заголовочный файл vector.h

```
#ifndef LAB14VECTOR_VECTOR_H
#define LAB14VECTOR_VECTOR_H
#include <stdio.h>
#include <malloc.h>
#include <stdint.h>
#include <stdbool.h>
#include <assert.h>
typedef struct vector {
    int *data; // указатель на элементы вектора
    size_t size; // размер вектора
    size_t capacity; // вместимость вектора
} vector;
#endif //LAB14VECTOR_VECTOR_H
```

Файл реализации vector.c

```
#include "vector.h"
//возвращает структуру-дескриптор вектор из n значений
vector createVector(size_t n) {
    vector vec;
    vec.data = malloc(n * sizeof(int));
    vec.size = 0;
    vec.capacity = n;

    if (vec.data == NULL) {
        fprintf(stderr, "bad alloc");
        exit(1);
    }

    vec.size = 0;
    vec.capacity = n;

    return vec;
}
//освобождает память, выделенную вектору
void deleteVector(vector *v) {
    free(v->data);
    v->size = 0;
    v->capacity = 0;
}
//удаляет элементы из контейнера, но не освобождает выделенную память
void clearVector(vector *v) {
    v->size = 0;
}
/*изменяет количество
памяти, выделенное под хранение элементов вектора. */
void reserveVector(vector *v, size_t newCapacity) {
    if (newCapacity == 0){
        v->data = NULL;
    }

    if (newCapacity < v->size){
        v->size = newCapacity;
    }

    v->capacity = newCapacity;
    v->data = (int *) realloc(v->data, newCapacity * sizeof(int));

    if (v->data == NULL){
        fprintf(stderr, "bad alloc");
        exit(1);
    }
```

```
}
//возвращает i-ый элемент вектора v.
int getVectorValue(vector *v, size_t i) {
    return v->data[i];
}
/*Добавляет элемент x в конец вектора v.
Если вектор заполнен, количество выделенной ему памяти увеличено в
2 раза, используя reserve. */
void pushBackVector(vector *v, const int x) {
    if (!v->capacity)
        reserveVector(v, 1);

    if (v->size >= v->capacity)
        reserveVector(v, v->capacity * 2);

    v->data[v->size++] = x;
}
/*освобождает память, выделенную под
неиспользуемые элементы.*/
void shrinkToFit(vector *v) {
    reserveVector(v, v->size);
}
//является ли вектор пустым
bool vector_isEmpty(const vector v) {
    return v.size == 0;
}
//является ли вектор полным
bool vector_isFull(const vector v) {
    return v.size == v.capacity;
}
//удаляет последний элемент из вектора.
void VectorpopBack(vector *v) {
    if (!v->data) {
        printf("Error: Failed access to vector's memory in VectorpopBack\n");
        return;
    }

    if (v->size == 0) {
        printf("Error: Vector is empty in VectorpopBack\n");
        return;
    }

    v->size--;
    v->data[v->size] = 0;
}

int* atVector(vector *v, size_t index) {
    if (!v->data) {
        printf("Error: Failed access to vector's memory in atVector\n");
        return NULL;
    }

    if (index >= v->size) {
        fprintf(stderr, "IndexError: a[index] is not exists");
        exit(1);
    }

    return v->data + index;
}

int* back(vector *v){
```

```
    if (!v->size){
        return &v->data[v->size];
    }

    return &v->data[v->size - 1];
}

int* front(vector *v) {
    if (!v->data) {
        fprintf(stderr, "Error accessing vector memory");
    }

    return v->data;
}
```

Тесты:

```
//Тесты функций
void test_pushBackVector_emptyVector() {

    vector v = createVector(5);

    pushBackVector(&v,10);

    printf("pushBack_emptyVector :%d\n", v.data[0]);

}

void test_pushBackVector_fullVector(){

    vector v = createVector(5);

    v.size = 5;
    v.data[0] = 20;
    v.data[1] = 20;
    v.data[2] = 20;
    v.data[3] = 20;
    v.data[4] = 20;

    pushBackVector(&v, 21);

    printf("\n pushBack_fullVector: %d\n", v.data[5]);

}

void test_popBack_notEmptyVector() {
    vector v = createVector(0);
    pushBackVector(&v, 10);
    assert(v.size == 1);
    VectorpopBack(&v);
    assert(v.size == 0);
    assert(v.capacity == 1);
}

void test_atVector_notEmptyVector() {

    vector v = createVector(5);

    v.size = 4;
    v.data[0] = 20;
    v.data[1] = 21;
    v.data[2] = 22;
    v.data[3] = 23;

    printf("\n atVector_notEmptyVector address: %x \n", atVector(&v, 2));
    printf("\n atVector_notEmptyVector value: %d \n", *atVector(&v, 2));
}

void test_atVector_requestToLastElement(){

    vector v = createVector(5);

    v.size = 3;
    v.data[0] = 20;
    v.data[1] = 20;
    v.data[2] = 20;
//Вывод в 16ричной с\с чтобы посмотреть нормально адрес
    printf("\n requestToLastElement value: %x \n", *atVector(&v, 2));
```

```
    printf("\n requestToLastElement address: %d \n", atVector(&v, 2));
}

void test_back_oneElementInVector(){

    vector v = createVector(5);

    v.size = 1;
    v.data[0] = 20;
//Вывод в 16ричной с\с чтобы посмотреть нормально адрес
    printf("\n back_oneElementInVector address: %x \n", back(&v));
    printf("\n back_oneElementInVector value: %d \n", *back(&v));
}

void test_front_oneElementInVector() {

    vector v = createVector(5);

    v.size = 1;
    v.data[0] = 20;
//Вывод в 16ричной с\с чтобы посмотреть нормально адрес
    printf("\n front_oneElementInVector address: %x \n", front(&v));
    printf("\n front_oneElementInVector value: %d \n", *front(&v));
}
int main() {

    test_popBack_notEmptyVector();
    test_pushBackVector_emptyVector();
    test_pushBackVector_fullVector();
    test_atVector_notEmptyVector();
    test_atVector_requestToLastElement();
    test_back_oneElementInVector();
    test_front_oneElementInVector();

    return 0;
}
```

Результаты тестирования:

```
C:\Users\Александр\CLionProjects\LAB14Vector\vector.exe
pushBack_emptyVector :10

 pushBack_fullVector: 21

 atVector_notEmptyVector address: 840e14b8

 atVector_notEmptyVector value: 22

 requestToLastElement address: 840e14d8

 requestToLastElement value: 20

 back_oneElementInVector address: 840e14f0

 back_oneElementInVector value: 20

 front_oneElementInVector address: 840e1510

 front_oneElementInVector value: 20

Process finished with exit code 0
```

Результаты выполнения:

```
Александр@DESKTOP-8QVSDRR MINGW64 ~/CLionProjects/LAB14Vector (master)
$ git log --stat -- vector.c
commit 6d655f16239c752f2886b9c6cf3a41564603e739 (HEAD -> master, origin/master)
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Feb 29 11:23:16 2024 +0300

    memory usage of vector

 vector.c | 16 ++++++++--------
 1 file changed, 8 insertions(+), 8 deletions(-)

commit e59e80f574e8691f7495c4b82f95338e4992c1b8
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Wed Feb 28 11:43:49 2024 +0300

    изменено:test_front_oneElementInVector была опечатка

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

commit ec50aeee3fa50a2273afd8f7f9116784f257a6df
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Wed Feb 28 11:18:34 2024 +0300

    refactoring
:...skipping...
commit 6d655f16239c752f2886b9c6cf3a41564603e739 (HEAD -> master, origin/master)
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Feb 29 11:23:16 2024 +0300

    memory usage of vector

 vector.c | 16 ++++++++--------
 1 file changed, 8 insertions(+), 8 deletions(-)

commit e59e80f574e8691f7495c4b82f95338e4992c1b8
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Wed Feb 28 11:43:49 2024 +0300

    изменено:test_front_oneElementInVector была опечатка

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

commit ec50aeee3fa50a2273afd8f7f9116784f257a6df
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Wed Feb 28 11:18:34 2024 +0300

    refactoring

 vector.c | 171 ++++++++++++++++++++++++++++++++++++++++++++++++++++++---------
 1 file changed, 147 insertions(+), 24 deletions(-)

commit d22595403db2780fc0891ee3d328e939870b90b5
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Feb 27 12:05:01 2024 +0300

    reserve 100%, isEmpty, isFull,pushBack

 vector.c | 45 +++++++++++++++++++++++++++++++++------------
 1 file changed, 33 insertions(+), 12 deletions(-)

commit fdcad63c7176832828db81fbd534289b20c1c999
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Mon Feb 26 17:47:26 2024 +0300

    Add delete, clear and shrinkToFit (reserve 50/50)

 vector.c | 54 +++++++++++++++++++++++++++++++++++-------------------------
 1 file changed, 35 insertions(+), 19 deletions(-)
```

```
commit fc7d8cb8fa75e475d85d463264a381921abe9543
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Sun Feb 25 18:51:27 2024 +0300

    Primer

 vector.c | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)

commit 5c415bff42e29b2e6d80d60e2664cb967261c8fd
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Sun Feb 25 18:49:40 2024 +0300

    first commit

 vector.c | 65 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 65 insertions(+)
```

Заголовочный файл vectorVoid.h:

```
#ifndef LAB14VECTOR_VECTORVOID_H
#define LAB14VECTOR_VECTORVOID_H
#include <limits.h>
#include <stdio.h>
#include <malloc.h>
#include <stdint.h>
#include <stdbool.h>
#include <assert.h>
#include <memory.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;
#endif //LAB14VECTOR_VECTORVOID_H
```

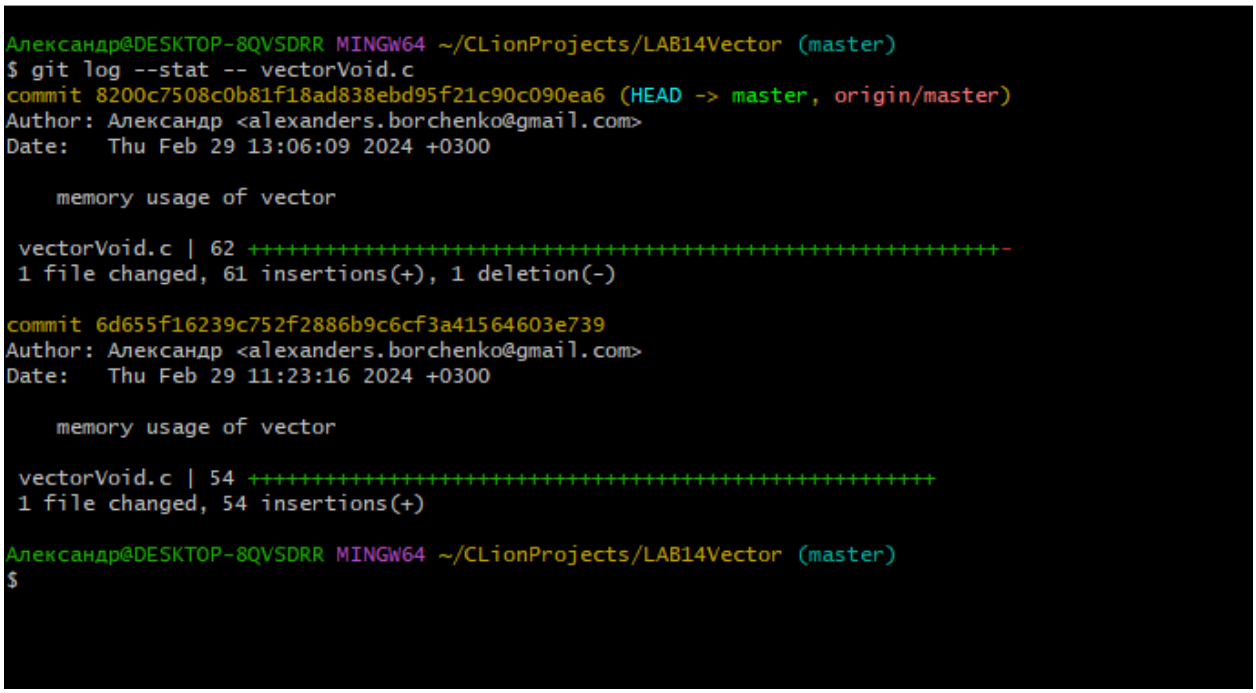
Файл реализации vectorVoid.c:

```
#include "vectorVoid.h"
//Создание вектора с baseTypeSize типом
vectorVoid createVectorV(size_t n, const size_t baseTypeSize) {
    vectorVoid vec;
    vec.data = malloc(n * baseTypeSize);
    vec.size = 0;
    vec.capacity = n;

    if (vec.data == NULL) {
        fprintf(stderr, "bad alloc");
        exit(1);
    }

    vec.size = 0;
    vec.capacity = n;
    vec.baseTypeSize = baseTypeSize;

    return vec;
}
/*изменяет количество
памяти, выделенное под хранение элементов вектора. */
void reserveVectorV(vectorVoid *v, size_t newCapacity) {
    if (newCapacity == 0){
        v->data = NULL;
    }

    if (newCapacity < v->size){
        v->size = newCapacity;
    }

    v->capacity = newCapacity;
    v->data = (int *) realloc(v->data, newCapacity * v->baseTypeSize);

    if (v->data == NULL){
        fprintf(stderr, "bad alloc");
        exit(1);
    }
}
/*освобождает память, выделенную под
неиспользуемые элементы.*/
```

```
void shrinkToFitV(vectorVoid *v) {
    reserveVectorV(v, v->size);
}
//удаляет элементы из контейнера, но не освобождает выделенную память
void clearVectorV(vectorVoid *v) {
    v->size = 0;
}
//освобождает память, выделенную вектору
void deleteVectorV(vectorVoid *v) {
    free(v->data);
    v->size = 0;
    v->capacity = 0;
    v->baseTypeSize = 0;
}
//является ли вектор пустым
bool vector_isEmptyV(const vectorVoid v) {
    return v.size == 0;
}
//является ли вектор полным
bool vector_isFullV(const vectorVoid v) {
    return v.size == v.capacity;
}
//записывает по адресу destination index-ый элемент вектора v.
void getVectorValueV(vectorVoid *v, size_t index, void *destination) {
    if (!v->data) {//Ошибка доступа к памяти
        fprintf(stderr, "Error accessing vector memory");
    }
    if (v->size <= index) {//индекс за пределами вектора
        fprintf(stderr, "index has gone beyond range of vector");
    }

    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 (!v->data) {//Ошибка доступа к памяти
        fprintf(stderr, "Error accessing vector memory");
    }
    if (v->size <= index) {//индекс за пределами вектора
        fprintf(stderr, "index has gone beyond range of vector");
    }

    char *destination = (char *) v->data + index * v->baseTypeSize;
    memcpy(destination, source, v->baseTypeSize);
}
//Добавляет
void pushBackVectorV(vectorVoid *v, void *source) {
    if (!v->capacity)
        reserveVectorV(v, 1);

    if (v->size >= v->capacity)
        reserveVectorV(v, v->capacity * 2);

    setVectorValueV(v, v->size++, source);
}
//удаляет
void popBackVectorV(vectorVoid *v) {
    if (!v->data) {
        fprintf(stderr, "Error: Failed access to vector's memory in
VectorpopBack\n");
    }
```

```
    if (v->size == 0) {
        fprintf(stderr,"Error: Vector is empty in VectorpopBack\n");
    }

    (v->size)--;
}
```

Результат выполнения:



Вывод: в ходе выполнения лабораторной работы я усовершенствовал свои навыки в создании библиотек, получил навыки работы с векторами на ЭВМ и с системой контроля версий git.