**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра «Информационные системы»**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: Линейные структуры данных. Динамические массивы и двусвязные списки.

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Сомова П.Д. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2021

**Цель работы.**

Изучение свойств и организация динамических массивов и двусвязных списков; получение практических навыков в работе с динамическими массивами и двусвязными списками; проведение сравнительной характеристики скорости вставки, получения и удаления элементов из них.

**Основные теоретические положения.**

В языке С++, за исключением класса vector, отсутствуют инструменты работы массивами переменной длины, т.е. с массивами, длина которых меняется во время работы программы.

Обойти данное ограничение позволяет т.н. динамическое выделение памяти, реализуемое в языках Си и C++ с помощью функций:

* malloc
* calloc
* realloc
* free

Первые 3 функции выделяют память и возвращают указатель на неё. Calloc, при этом ещё и инициализирует память нулями, а realloc - данными, располагающимися в памяти по адресу, передаваемому функции в качестве аргумента. Функция free, же, освобождает выделенную ранее память.

В языке c++ присутствует, также, и способ работы с динамически выделяемой памятью с помощью операторов new и delete. Однако, данный способ медленнее и менее функционален, чем унаследованный от языка си.

Кроме динамических массивов в языке c++ возможно использование для тех же целей и линейные списки.

Односвязный (однонаправленный) список представляет из объектов, содержащих значение и указатель на следующий подобный объект. Двусвязный (двунаправленный) список же содержит, кроме того и указатель на предыдущий элемент. Очевидно, что наиболее удобный способ реализации линейных списков в языках си/c++ представляют собой структуры.

Пример подобной структуры:

**struct** list

**{**

**list** \*head; //Указатель на предыдущий элемент

**int** value; //Значение элемента

**list** \*tail; //Указатель на следующий элемент

**};**

**Постановка задачи.**

Необходимо реализовать программу, которая выполняет следующие действия.

1. Формирование целочисленного одномерного массива размерности N, где:

a) пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;

в) массив считывается с файла, N определяется как количество элементов массива в файле.

2. Определение скорости создания динамического массива п. 1.

3. Вставка, удаление и получение элемента массива. Удаление и получение элемента необходимо реализовать по индексу и по значению.

4. Определение скорости вставки, удаления и получения элемента массива п. 3.

5. Формирование двусвязного списка размерности N, где:

a) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы списка, N определяется автоматически по количеству введенных элементов;

в) список считывается с файла, N определяется как количество элементов списка в файле.

6. Определение скорости создания двусвязного списка п. 5.

7. Вставка, удаление и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.

8. Определение скорости вставки, удаление и получения элемента двусвязного списка п. 7.

Должна быть возможность запуска каждого пункта многократно, если есть возможность (если в списке/массиве нет элементов, то нельзя ничего удалить и об этом нужно сообщить пользователю). Необходимо сравнить результаты. Для этого пункты 1–4 и 5–8 должны принимать одинаковые значения.Сделать вывод по проделанной работе.

**Выполнение работы.**

Для решения поставленной задачи была написана программа на языке C++. Итоговый код программы представлен в приложении А.

Было проведено тестирование программы с использование различных ЭВМ и компиляторов. Результаты тестирования представлены в приложении Б. При этом были получены результаты, соответствующие расчётным.

**Выводы.**

Осуществление операций над динамическим массивом почти всегда быстрее, чем над линейным списком. Исключением является осуществление операций над массивом/списком большой длины, не требующих перебора всего списка/массива.

Приложение А

Полный код программы

#include <iostream>  
#include <fstream>  
#include <chrono>  
  
typedef std :: chrono :: time\_point<std :: chrono :: high\_resolution\_clock> timestamp**;**using namespace std**;**int \* createRandomArrSize ( struct Array \* arr )**;**int \* createArrayWithoutSize ( struct Array \* arr )**;**int \* createFromFile ( struct Array \* arr )**;**int \* addElementToArray ( struct Array \* arr**,** int element )**;**int \* deleteElementIndex ( struct Array \* arr**,** int index )**;**int \* deleteElementValue ( struct Array \* arr**,** int value )**;**int getElement ( int \* arr**,** int index )**;**int \* doubleUpArray ( struct Array \* arr )**;**int \* doubleUpArray ( int \* arr**,** int size )**;**int \* shrinkArray ( int \* arr**,** int newSize )**;**int \* shiftArray ( int \* arr**,** int size**,** int startIndex )**;**void printArray ( struct Array \* arr )**;**void printMenu ()**;**void chooseMenu (struct Array \* arr**,** int menuCase )**;**void printTime ( timestamp**,** timestamp**,** string )**;**struct Array {  
 int \* data**;** int size**;** int length**;**}**;**struct List {  
 int data**;** List \*head**;** List \*tail**;** int length**;**}**;**int main() {  
  
 setlocale(LC\_ALL**,** "Russian")**;** int menuCase**;** struct Array \*arr = new Array()**;** while (true) {  
  
 printMenu()**;** cin >> menuCase**;** cout << "\n"**;** switch (menuCase) {  
 case **7**:  
 delete[] arr**;** return **0;** break**;** default:  
 chooseMenu(arr**,** menuCase)**;** break**;** }  
  
 }  
}  
  
void printMenu () {  
  
 cout << "---Массив---:\n"  
 << "1 - Создать массив автоматически\n"  
 << "2 - Создать массив с клавиатуры\n"  
 << "3 - Создать массив с файла\n"  
 << "4 - Добавить элемеент в массив\n"  
 << "5 - Удалить элемент из массива\n"  
 << "6 - Получить элемент из массива\n\n"  
 // << "===Double Linked List===\n"  
 // << "7 - Create double linked list automaticly\n"  
 // << "8 - Create double linked list from input\n"  
 // << "9 - Create double linked list from file\n"  
 // << "10 - Add element to list\n"  
 // << "11 - Delete element from list\n"  
 // << "12 - Get element from list\n"  
 << "7 - Выход\n"  
 << "Введите пункт: "**;**}  
  
void chooseMenu ( struct Array \* arr**,** int menuCase ) {  
  
 switch (menuCase) {  
  
 case **1**: {  
 cout << "Введите размер массива: "**;** cin >> arr->size**;** arr->length = arr->size - **1;** arr->data = createRandomArrSize(arr)**;** cout << "Массив создан: "**;** printArray(arr)**;** break**;** }  
 case **2**: {  
 std::cout << "Введите элементы массива: \n"**;** arr->data = createArrayWithoutSize(arr)**;** cout << "Массив создан: "**;** printArray(arr)**;** break**;** }  
 case **3**: {  
 arr->data = createFromFile(arr)**;** cout << "Массив создан: "**;** printArray(arr)**;** break**;** }  
 case **4**: {  
 int newEl**;** cout << "Введите элемент для добавления в массив: "**;** cin >> newEl**;** arr->data = addElementToArray(arr**,** newEl)**;** printArray(arr)**;** break**;** }  
 case **5**: {  
 int toDelete**,** type**;** cout << "Выберите вариант: 1 - по индексу; 2 - по значению\n"**;** cin >> type**;** switch (type) {  
 case **1**:  
 cout << "Введите индекс для удаления: "**;** cin >> toDelete**;** arr->data = deleteElementIndex(arr**,** toDelete)**;** break**;** case **2**:  
 cout << "Введите значение для удаления: "**;** cin >> toDelete**;** arr->data = deleteElementValue(arr**,** toDelete)**;** break**;** }  
 printArray(arr)**;** break**;** }  
 case **6**: {  
 int index**;** cout << "Введите индекс для получения: "**;** cin >> index**;** cout << "Индекс значения: " << getElement(arr->data**,** index) << "\n"**;** break**;** }  
 }  
}  
  
  
  
  
  
  
/\*  
 \* Array  
 \*/  
int \* createRandomArrSize ( struct Array \* arr ) {  
  
 int \* newArr = new int[arr->size]**;** timestamp startTime**,** endTime**;** startTime = std :: chrono :: high\_resolution\_clock :: now()**;** for ( int i = **0;** i < arr->size**;** i++ ) {  
 newArr[i] = rand() % **100;** }  
  
 endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "Array created: ")**;** return newArr**;**}  
  
int \* createArrayWithoutSize ( struct Array \* arr ) {  
  
 int size = **2;** int \* newArr = new int[size]**;** int num**,** index = **0;** bool end = false**;** arr->length = size - **1;** arr->size = size**;** timestamp startTime**,** endTime**;** startTime = std :: chrono :: high\_resolution\_clock :: now()**;** while ( !end ) {  
  
 cout << "Введите " << index << " элемент ( или -1, чтобы прекратить ввод ): "**;** cin >> num**;** if ( num == -**1** ) {  
 end = true**;** break**;** } else if ( index < size ) {  
 newArr[index] = num**;** index++**;** } else {  
 index = size**;** int \* temp = newArr**;** newArr = doubleUpArray(temp**,** size)**;** newArr[index] = num**;** size = size \* **2;** index++**;** }  
  
 }  
  
 endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "Array created ")**;** arr->length = index - **1;** return newArr**;**}  
  
int \* createFromFile ( struct Array \* arr ) {  
  
 ifstream file**;** int index = **0,** size = **2;** int \* newArr = new int[size]**;** timestamp startTime**,** endTime**;** startTime = std :: chrono :: high\_resolution\_clock :: now()**;** file.open("array.txt")**;** if ( file.is\_open() ) {  
  
 string line**;** while ( getline(file**,** line) ) {  
  
 int num = stoi(line)**;** if (index < size) {  
 newArr[index] = num**;** index++**;** } else {  
 index = size**;** int\* temp = newArr**;** newArr = doubleUpArray(temp**,** size)**;** newArr[index] = num**;** size = size \* **2;** index++**;** }  
  
 }  
  
 }  
  
 endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "Array creation ")**;** arr->length = index - **1;** return newArr**;**}  
  
  
int \* addElementToArray ( struct Array \* arr**,** int element ) {  
  
 timestamp startTime**,** endTime**;** startTime = std :: chrono :: high\_resolution\_clock :: now()**;** if ( arr->length < arr->size ) {  
 arr->length++**;** arr->data[arr->length] = element**;** endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "Добавлен элемент ")**;** return arr->data**;** } else {  
 int \* newArr = doubleUpArray(arr)**;** arr->length++**;** newArr[arr->length] = element**;** endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "Добавлен элемент ")**;** return newArr**;** }  
}  
  
  
int \* deleteElementIndex ( struct Array \* arr**,** int index ) {  
  
 if ( arr->length == -**1** ) {  
 cout << "Массив пустой\n"**;** return arr->data**;** }  
  
 if ( index == arr->length ) {  
 arr->length--**;** arr->size = arr->length**;** return shrinkArray(arr->data**,** arr->length)**;** } else {  
 arr->length--**;** arr->size = arr->length**;** return shiftArray(arr->data**,** arr->length**,** index)**;** }  
  
}  
  
int \* deleteElementValue ( struct Array \* arr**,** int value ) {  
  
 int index**;** for (int i = **0;** i < arr->length**;** i++) {  
 if ( arr->data[i] == value ) {  
 return deleteElementIndex(arr**,** i)**;** }  
 }  
  
 cout << "Значение удалено\n"**;** return arr->data**;**}  
  
int getElement ( int \* arr**,** int index ) {  
  
 cout << "Элемент был получен за 0 ms: "**;** return arr[index]**;**}  
  
  
//доп к массивам  
  
int \* doubleUpArray ( struct Array \* arr ) {  
  
 int newSize = arr->size \* **2;** int \* newArr = new int[newSize]**;** timestamp startTime**,** endTime**;** startTime = std :: chrono :: high\_resolution\_clock :: now()**;** for ( int i = **0;** i <= arr->length**;** i++ ) {  
 newArr[i] = arr->data[i]**;** }  
  
 endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "")**;** arr->size = newSize**;** return newArr**;**}  
  
int \* doubleUpArray ( int \* arr**,** int size ) {  
  
 int newSize = size \* **2;** int \* newArr = new int[newSize]**;** timestamp startTime**,** endTime**;** startTime = std :: chrono :: high\_resolution\_clock :: now()**;** for ( int i = **0;** i <= size**;** i++ ) {  
 newArr[i] = arr[i]**;** }  
  
 endTime = std :: chrono :: high\_resolution\_clock :: now()**;** printTime(startTime**,** endTime**,** "")**;** return newArr**;**}  
  
int \* shrinkArray ( int \* arr**,** int newSize ) {  
  
 int \* newArr = new int[newSize + **1**]**;** for ( int i = **0;** i < newSize + **1;** i++ ) {  
 newArr[i] = arr[i]**;** }  
  
 return newArr**;**}  
  
int \* shiftArray ( int \* arr**,** int size**,** int startIndex ) {  
  
 int \* newArr = new int[size]**;** for ( int i = **0;** i < startIndex**;** i++ ) {  
 newArr[i] = arr[i]**;** }  
  
 for ( int i = startIndex**;** i < size + **1;** i++ ) {  
 newArr[i] = arr[i + **1**]**;** }  
  
 return newArr**;**}  
  
void printArray ( struct Array \* arr ) {  
  
 for ( int i = **0;** i <= arr->length**;** i++ ) {  
 cout << arr->data[i] << " "**;** }  
  
 cout << "\n\n"**;**}  
  
void printTime ( timestamp startTime**,** timestamp endTime**,** string type ) {  
  
 cout << type << "ended in: "  
 << std :: chrono :: duration\_cast <std :: chrono :: microseconds> (endTime - startTime).count()  
 << " microseconds\n"**;**}  
  
/\*  
 \* List  
 \*/  
  
/\*  
void createRandomeList ( List \* list ) {  
  
 timestamp startTime, endTime;  
 startTime = chrono :: high\_resolution\_clock :: now();  
  
 for ( int i = 0; i < list->length; i++ ) {  
 push(list, rand() % 100);  
 }  
  
 endTime = chrono :: high\_resolution\_clock :: now();  
 printTime(startTime, endTime, "List created ");  
  
}  
  
void populateListUndefined ( List \* list ) {  
  
 int num, index = 0;  
 bool end = false;  
  
 timestamp startTime, endTime;  
 startTime = chrono :: high\_resolution\_clock :: now();  
  
 while ( !end ) {  
  
 cout << "Введите " << index << " элемент ( или -1, чтобы прекратить ввод ): ";  
  
 cin >> num;  
  
 if ( num == -1 ) {  
 end = true;  
 break;  
 } else {  
 index++;  
 push(list, num);  
 }  
  
 }  
  
 endTime = chrono :: high\_resolution\_clock :: now();  
 printTime(startTime, endTime, "List createdн ");  
  
}  
\*/

}

Приложение Б

результаты тестирования

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:1

Введите размер массива:5

Array created: ended in: 0 microseconds

Массив создан: 41 67 34 0 69

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:2

Введите элементы массива:

Введите 0 элемент ( или -1, чтобы прекратить ввод ):1

Введите 1 элемент ( или -1, чтобы прекратить ввод ):2

Введите 2 элем

ент ( или -1, чтобы прекратить ввод ):3

ended in: 0 microseconds

Введите 3 элемент ( или -1, чтобы прекратить ввод ):4

Введите 4 элемент ( или -1, чтобы прекратить ввод ):5

ended in: 0 mi

croseconds

Введите 5 элемент ( или -1, чтобы прекратить ввод ):-1

Array created ended in: 7629723 microseconds

Массив создан: 1 2 3 4 5

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:3

ended in: 0 microseconds

ended in: 0 microseconds

Array creation ended in: 4987 microseconds

Массив создан: 2 5 3 4 6 48

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:4

Введите элемент для добавления в массив:52

ended in: 0 microseconds

Добавлен элемент ended in: 1995 microseconds

2 5 3 4 6 48 52

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:5

Выберите вариант: 1 - по индексу; 2 - по значению

1

Введите индекс для удаления:0

5 3 4 6 48 52

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:5

Выберите вариант: 1 - по индексу; 2 - по значению

2

Введите значение для удаления:4

5 3 6 48 52

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:6

Введите индекс для получения:2

Индекс значения: Элемент был получен за 0 ms: 6

---Массив---:

1 - Создать массив автоматически

2 - Создать массив с клавиатуры

3 - Создать массив с файла

4 - Добавить элемеент в массив

5 - Удалить элемент из массива

6 - Получить элемент из массива

7 - Выход

Введите пункт:7

Process finished with exit code 0