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 _____ Балашевич К.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 C++, 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:

а) 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:

а) 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

```
////////////////////////////////////////////////////////////////
//                          ВНИМАНИЕ                          //
////////////////////////////////////////////////////////////////
//  Для компиляции под ОС Windows с использованием компилятора  //
//            MSVC раскомментируйте следующую строку:           //
#define CLEARSCREEN "cls"                                       //
////////////////////////////////////////////////////////////////
//    Для компиляции под POSIX-совместимой ОС или ОС Windows с  //
// использованием компилятора GCC/G++ раскомментируйте следующую //
//                           строку:                            //
//#define CLEARSCREEN "clear"                                   //
////////////////////////////////////////////////////////////////

#include <iostream>
#include <fstream>
#include <limits>
#include <climits>
#include <chrono>

struct list
  {
    int value;
    list *prev;
    list *next;
  };

struct listAndArray
  {
    int listSize;
    list *List;
    int *Array;
  };

listAndArray* createListAndArray ();
int getNumericAnswer (int minRange, int maxRange, const char
*Question);
listAndArray* mainMenu (listAndArray *ListAndArray);
listAndArray* deleteListAndArray (listAndArray *ListAndArray);
void printList (listAndArray *ListAndArray);
void printArray (listAndArray *ListAndArray);
```

```
void insertElementToList (listAndArray *ListAndArray, int position,
int value);
void insertElementToArray (listAndArray *ListAndArray, int position,
int value);
void  deleteElementFromListByPosition  (listAndArray  *ListAndArray,
int position);
void  deleteElementFromArrayByPosition  (listAndArray  *ListAndArray,
int position);
void  deleteElementFromListByValue  (listAndArray  *ListAndArray,  int
value);
void deleteElementFromArrayByValue (listAndArray *ListAndArray, int
value);
listAndArray* readListFromTerm ();
listAndArray* readListFromFile ();
listAndArray* generateRandomList ();
listAndArray* insertElement (listAndArray *ListAndArray);
listAndArray* deleteElement (listAndArray *ListAndArray);
void printElement (listAndArray *ListAndArray);

int main ()
  {
    setlocale (0, "");
    srand (time (0));
    listAndArray *ListAndArray=NULL;
    ListAndArray=mainMenu (ListAndArray);
    deleteListAndArray (ListAndArray);
    return 0;
  }

listAndArray* createListAndArray ()
  {
    listAndArray     *ListAndArray=(listAndArray*)malloc     (sizeof
(listAndArray));
    ListAndArray->List=NULL;
    ListAndArray->Array=NULL;
    ListAndArray->listSize=0;
    return ListAndArray;
  }

int  getNumericAnswer  (int  minRange,  int  maxRange,  const  char
*Question)
  {
    int answer;
    std::cout << Question;
    std::cin >> answer;
```

```
    if ((std::cin.fail ())||(answer<minRange)||(answer>maxRange))
      {
        std::cin.clear ();
        std::cin.ignore   (std::numeric_limits<std::streamsize>::max
(), '\n');
        std::cout << "Некорректный ввод!" << std::endl;
        answer=getNumericAnswer (minRange, maxRange, Question);
        return answer;
      }
    else
      {
        std::cin.ignore   (std::numeric_limits<std::streamsize>::max
(), '\n');
        return answer;
      }
  }

listAndArray* mainMenu (listAndArray *ListAndArray)
  {
    system (CLEARSCREEN);
    std::cout << "Главное меню:\n\nТекущее состояние массива:" <<
std::endl;
    printArray (ListAndArray);
    std::cout << std::endl << "\nТекущее состояние списка:" <<
std::endl;
    printList(ListAndArray);
    std::cout << std::endl << "\n1) Сформировать список и массив
случайным обра"
                 "зом.\n2) Считать список и массив из терминала.\n3)
Считать сп"
                 "исок и массив из файла.\n4) Вставить элемент в
список и масси"
                 "в на произвольную позицию.";
    bool isListEmpty;
    if (ListAndArray!=NULL)
      {
        std::cout << "\n5) Удалить произвольный элемент из списка и
массива.\n6"
                     ") Вывести значение произвольного элемента
массива.\n0) Вы"
                     "йти из программы.\n" << std::endl;
        isListEmpty=0;
      }
    else
      {
```

```
        std::cout << "\n0) Выйти из программы.\n" << std::endl;
        isListEmpty=1;
      }
    switch (getNumericAnswer (0, isListEmpty?4:6, "Введите номер
желаемого варианта: "))
      {
        case 1:
          ListAndArray=deleteListAndArray (ListAndArray);
          ListAndArray=generateRandomList ();
          ListAndArray=mainMenu (ListAndArray);
          break;
        case 2:
          ListAndArray=deleteListAndArray (ListAndArray);
          ListAndArray=readListFromTerm ();
          ListAndArray=mainMenu (ListAndArray);
          break;
        case 3:
          ListAndArray=deleteListAndArray (ListAndArray);
          ListAndArray=readListFromFile ();
          ListAndArray=mainMenu (ListAndArray);
          break;
        case 4:
          ListAndArray=insertElement (ListAndArray);
          ListAndArray=mainMenu (ListAndArray);
          break;
        case 5:
          ListAndArray=deleteElement (ListAndArray);
          ListAndArray=mainMenu (ListAndArray);
          break;
        case 6:
          printElement (ListAndArray);
          ListAndArray=mainMenu (ListAndArray);
          break;
        default:
          break;
      }
    return ListAndArray;
  }

listAndArray* deleteListAndArray (listAndArray *ListAndArray)
  {
    if (ListAndArray!=NULL)
      {
        while (ListAndArray->List!=NULL)
          {
```

```
            list *Buffer;
            Buffer=ListAndArray->List->next;
            free (ListAndArray->List);
            ListAndArray->List=Buffer;
          }
        if (ListAndArray->Array!=NULL)
          free (ListAndArray->Array);
        free (ListAndArray);
      }
    return NULL;
  }

void printList (listAndArray *ListAndArray)
  {
    if (ListAndArray==NULL)
      std::cout << "Список пуст.";
    else
      {
        list* List=ListAndArray->List;
        do
          {
            std::cout << List->value << " ";
            List=List->next;
          }
        while (List!=NULL);
      }
  }

void printArray (listAndArray *ListAndArray)
  {
    if (ListAndArray==NULL)
      std::cout << "Массив пуст.";
    else
      {
        for (int i=0; i<ListAndArray->listSize; i++)
          std::cout << *(ListAndArray->Array+i) << " ";
      }
  }

void insertElementToList (listAndArray *ListAndArray, int position,
int value)
  {
    list *List=ListAndArray->List;
    list *Element=(list*)malloc (sizeof (list));
    Element->value=value;
```

```
    if (!(position-1))
      {
        Element->next=List;
        Element->prev=NULL;
        if (List!=NULL)
          List->prev=Element;
        ListAndArray->List=Element;
      }
    else
      {
        for (int i=1; i<position-1; i++)
            List=List->next;
        Element->next=List->next;
        List->next=Element;
        Element->prev=List;
        if (Element->next!=NULL)
          Element->next->prev=Element;
      }
  }

void insertElementToArray (listAndArray *ListAndArray, int position,
int value)
  {
    ListAndArray->Array=(int*)realloc          (ListAndArray->Array,
(ListAndArray->listSize+1)*sizeof(int));
    for (int i=ListAndArray->listSize; i>position-1; i--)
      *(ListAndArray->Array+i)=*(ListAndArray->Array+i-1);
    *(ListAndArray->Array+position-1)=value;
  }

void  deleteElementFromListByPosition  (listAndArray  *ListAndArray,
int position)
  {
    list *List=ListAndArray->List;
    for (int i=1; i<position; i++)
      List=List->next;
    if (List->prev!=NULL)
      List->prev->next=List->next;
    else
      ListAndArray->List=List->next;
    if (List->next!=NULL)
      List->next->prev=List->prev;
    free(List);
  }
```

```
void deleteElementFromArrayByPosition (listAndArray *ListAndArray,
int position)
  {
    for (int i=position-1; i<ListAndArray->listSize; i++)
      *(ListAndArray->Array+i)=*(ListAndArray->Array+i+1);
    ListAndArray->Array=(int*)realloc          (ListAndArray->Array,
(ListAndArray->listSize-1)*sizeof(int));
  }

void deleteElementFromListByValue (listAndArray *ListAndArray, int
value)
  {
    list *List=ListAndArray->List;
    list *Buffer;
    while (List!=NULL)
      {
        Buffer=List->next;
        if (List->value==value)
          {
            if (List->prev!=NULL)
              List->prev->next=List->next;
            else
              ListAndArray->List=List->next;
            if (List->next!=NULL)
              List->next->prev=List->prev;
            free(List);
          }
        List=Buffer;
      }
  }

void deleteElementFromArrayByValue (listAndArray *ListAndArray, int
value)
  {
    for (int i=0; i<ListAndArray->listSize;)
      if (*(ListAndArray->Array+i)==value)
        {
          for (int j=i; j<ListAndArray->listSize-1; j++)
            *(ListAndArray->Array+j)=*(ListAndArray->Array+j+1);
          ListAndArray->Array=(int*)realloc    (ListAndArray->Array,
(ListAndArray->listSize-1)*sizeof(int));
          ListAndArray->listSize--;
        }
      else
        i++;
```

```
  }

listAndArray* readListFromTerm ()
  {
    std::cout << "Введите элементы списка/массива в одну строку
через пробел: ";
    listAndArray *ListAndArray=createListAndArray ();
    char Buffer;
    Buffer=std::cin.get();
    std::chrono::system_clock::time_point             listTimeBegin,
listTimeEnd, arrayTimeBegin, arrayTimeEnd;
    while (Buffer!='\n')
      {
        while (!(Buffer-32))
          Buffer=std::cin.get();
        int value=0;
        bool isNegative=0;
        if (Buffer==45)
          {
            isNegative=1;
            Buffer=std::cin.get();
          }
        while (Buffer<58&&Buffer>47)
          {
            value=value*10+(Buffer-48);
            Buffer=std::cin.get();
          }
        if (isNegative)
          value=value*(-1);
        if (std::cin.fail()||(Buffer!=32&&Buffer!='\n'))
          {
            std::cin.clear();

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
            std::cout << "Некорректный ввод!" << std::endl;
            ListAndArray=deleteListAndArray(ListAndArray);
            ListAndArray=readListFromTerm();
            return ListAndArray;
          }
        listTimeBegin=std::chrono::system_clock::now();
        insertElementToList(ListAndArray,  ListAndArray->listSize+1,
value);
        listTimeEnd=std::chrono::system_clock::now();
        arrayTimeBegin=std::chrono::system_clock::now();
```

```
        insertElementToArray(ListAndArray, ListAndArray->listSize+1,
value);
        arrayTimeEnd=std::chrono::system_clock::now();
        ListAndArray->listSize++;
        while (!(Buffer-32))
          Buffer=std::cin.get();
      }
    if (!ListAndArray->listSize)
        ListAndArray=deleteListAndArray(ListAndArray);
    std::cout << "\nНа осуществеление операции над списком было
затрачено " <<

std::chrono::duration_cast<std::chrono::nanoseconds>(listTimeEnd-
listTimeBegin).count()
                << " наносекунд.\nНа осуществеление операции над
массивом было"
                "            затрачено            "            <<
std::chrono::duration_cast<std::chrono::nanoseconds>(arrayTimeEnd-
arrayTimeBegin).count()
                << " наносекунд.";
    std::cout << "\nДля возврата в меню нажмите \"Enter\".";
    std::cin.get();
    return ListAndArray;
  }

listAndArray* readListFromFile ()
  {
    char fileName[261];
    std::cout << std::endl << "Введите имя (или путь) файла: ";
    std::cin.getline (fileName, 261);
    std::ifstream File;
    File.open (fileName);
    listAndArray *ListAndArray=createListAndArray ();
    std::chrono::system_clock::time_point             listTimeBegin,
listTimeEnd, arrayTimeBegin, arrayTimeEnd;
    if (File.is_open())
      {

        while (!File.eof())
          {
            char *buffer=(char*)malloc (sizeof(char)*20);
            char *bufferTwo=buffer;
            File.getline(buffer, 20);
            int value=0;
            bool isNegative=0;
```

```
            if (*buffer==45)
              {
                isNegative=1;
                buffer++;
              }
            while (*buffer!='\0')
              {
                value=value*10+(*buffer-48);
                buffer++;
              }
            if (isNegative) value=value*(-1);
            free(bufferTwo);
            listTimeBegin=std::chrono::system_clock::now();
            insertElementToList     (ListAndArray,     ListAndArray-
>listSize+1, value);
            listTimeEnd=std::chrono::system_clock::now();
            arrayTimeBegin=std::chrono::system_clock::now();
            insertElementToArray    (ListAndArray,    ListAndArray-
>listSize+1, value);
            arrayTimeEnd=std::chrono::system_clock::now();
            ListAndArray->listSize++;
          }
        File.close();
      }
    else
      {
        ListAndArray=deleteListAndArray (ListAndArray);
        std::cout << "Некорректный ввод!" << std::endl;
        ListAndArray=readListFromFile ();
      }
    if (!ListAndArray->listSize)
        ListAndArray=deleteListAndArray (ListAndArray);
    std::cout  <<  "\nНа осуществление операции над списком было
затрачено " <<

std::chrono::duration_cast<std::chrono::nanoseconds>(listTimeEnd-
listTimeBegin).count()
                <<  "  наносекунд.\nНа осуществление операции над
массивом было"
                "             затрачено             "             <<
std::chrono::duration_cast<std::chrono::nanoseconds>(arrayTimeEnd-
arrayTimeBegin).count()
                << " наносекунд.";
    std::cout << "\nДля возврата в меню нажмите \"Enter\".";
    std::cin.get();
```

```
    return ListAndArray;
  }

listAndArray* generateRandomList ()
  {
    listAndArray *ListAndArray=createListAndArray ();
    int Size=getNumericAnswer(0, INT_MAX, "Введите число элементов
в списке/массиве: ");
    std::chrono::system_clock::time_point             listTimeBegin,
listTimeEnd, arrayTimeBegin, arrayTimeEnd;
    for (int i=0; i<Size; i++)
      {
        int value=rand()%100;
        listTimeBegin=std::chrono::system_clock::now();
        insertElementToList(ListAndArray,  ListAndArray->listSize+1,
value);
        listTimeEnd=std::chrono::system_clock::now();
        arrayTimeBegin=std::chrono::system_clock::now();
        insertElementToArray(ListAndArray, ListAndArray->listSize+1,
value);
        arrayTimeEnd=std::chrono::system_clock::now();
        ListAndArray->listSize++;
      }
    if (!ListAndArray->listSize)
        ListAndArray=deleteListAndArray(ListAndArray);
    std::cout  <<  "\nНа осуществление операции над списком было
затрачено " <<

std::chrono::duration_cast<std::chrono::nanoseconds>(listTimeEnd-
listTimeBegin).count()
                <<  "  наносекунд.\nНа осуществление операции над
массивом было"
                "             затрачено             "             <<
std::chrono::duration_cast<std::chrono::nanoseconds>(arrayTimeEnd-
arrayTimeBegin).count()
                << " наносекунд.";
    std::cout << "\nДля возврата в меню нажмите \"Enter\".";
    std::cin.get();
    return ListAndArray;
  }

listAndArray* insertElement (listAndArray *ListAndArray)
  {
    if (ListAndArray==NULL)
        ListAndArray=createListAndArray ();
```

```
    std::chrono::system_clock::time_point             listTimeBegin,
listTimeEnd, arrayTimeBegin, arrayTimeEnd;
    int   position=getNumericAnswer   (1,   ListAndArray->listSize+1,
"Введите номер позиции добавляемого элемента в списке/массиве: ");
    int value=getNumericAnswer (INT_MIN, INT_MAX, "Введите значение
добавляемого элемента: ");
    listTimeBegin=std::chrono::system_clock::now();
    insertElementToList (ListAndArray, position, value);
    listTimeEnd=std::chrono::system_clock::now();
    arrayTimeBegin=std::chrono::system_clock::now();
    insertElementToArray (ListAndArray, position, value);
    arrayTimeEnd=std::chrono::system_clock::now();
    ListAndArray->listSize++;
    std::cout  <<  "\nНа осуществеление операции над списком было
затрачено " <<

std::chrono::duration_cast<std::chrono::nanoseconds>(listTimeEnd-
listTimeBegin).count()
                <<  "  наносекунд.\nНа осуществеление операции над
массивом было"
                "             затрачено             "             <<
std::chrono::duration_cast<std::chrono::nanoseconds>(arrayTimeEnd-
arrayTimeBegin).count()
                << " наносекунд.";
    std::cout << "\nДля возврата в меню нажмите \"Enter\".";
    std::cin.get();
    return ListAndArray;
  }

listAndArray* deleteElement (listAndArray *ListAndArray)
  {
    std::cout << "Удалить элемент по значению или по номеру позиции
в массиве/с"
                "писке?\n1)  По  значению.\n2)  По  позиции."  <<
std::endl;
    bool   byPosition=getNumericAnswer   (1,   2,   "Введите   номер
жедаемого варианта: ")-1;
    std::chrono::system_clock::time_point             listTimeBegin,
listTimeEnd, arrayTimeBegin, arrayTimeEnd;
    if (byPosition)
      {
        int position=getNumericAnswer (1, ListAndArray->listSize+1,
"Введите номер позиции удаляемого элемента в списке/массиве: ");
        listTimeBegin=std::chrono::system_clock::now();
        deleteElementFromListByPosition (ListAndArray, position);
```

```
        listTimeEnd=std::chrono::system_clock::now();
        arrayTimeBegin=std::chrono::system_clock::now();
        deleteElementFromArrayByPosition (ListAndArray, position);
        arrayTimeEnd=std::chrono::system_clock::now();
        ListAndArray->listSize--;
        if (ListAndArray->List==NULL)
            ListAndArray=deleteListAndArray(ListAndArray);
      }
    else
      {
        int value=getNumericAnswer (INT_MIN, INT_MAX, "Введите
значение удаляемого элемента: ");
        listTimeBegin=std::chrono::system_clock::now();
        deleteElementFromListByValue (ListAndArray, value);
        listTimeEnd=std::chrono::system_clock::now();
        arrayTimeBegin=std::chrono::system_clock::now();
        deleteElementFromArrayByValue (ListAndArray, value);
        arrayTimeEnd=std::chrono::system_clock::now();
        if (ListAndArray->List==NULL)
            ListAndArray=deleteListAndArray(ListAndArray);
      }
    std::cout << "\nНа осуществление операции над списком было
затрачено " <<

std::chrono::duration_cast<std::chrono::nanoseconds>(listTimeEnd-
listTimeBegin).count()
                << " наносекунд.\nНа осуществление операции над
массивом было"
                "            затрачено            "            <<
std::chrono::duration_cast<std::chrono::nanoseconds>(arrayTimeEnd-
arrayTimeBegin).count()
                << " наносекунд.";
    std::cout << "\nДля возврата в меню нажмите \"Enter\".";
    std::cin.get();
    return ListAndArray;
  }

void printElement (listAndArray *ListAndArray)
  {
    std::cout << "Вывести элемент по значению или по номеру позиции
в массиве/с"
                "писке?\n1) По значению.\n2) По позиции." <<
std::endl;
    bool byPosition=getNumericAnswer (1, 2, "Введите номер
жедаемого варианта: ")-1;
```

```
    std::chrono::system_clock::time_point             listTimeBegin,
listTimeEnd, arrayTimeBegin, arrayTimeEnd;
    if (byPosition)
      {
        int position=getNumericAnswer (1, ListAndArray->listSize+1,
"Введите номер позиции выводимого элемента в списке/массиве: ");
        std::cout << "Элемент на позиции " << position << " в
списке: ";
        listTimeBegin=std::chrono::system_clock::now();
        list *List=ListAndArray->List;
        for (int i=1; i<position; i++)
          List=List->next;
        listTimeEnd=std::chrono::system_clock::now();
        std::cout << List->value << std::endl;
        std::cout << "Элемент на позиции " << position << " в
массиве: ";
        int value;
        arrayTimeBegin=std::chrono::system_clock::now();
        value=*(ListAndArray->Array+position-1);
        arrayTimeEnd=std::chrono::system_clock::now();
        std::cout << value;
      }
    else
      {
        int  value=getNumericAnswer  (INT_MIN,  INT_MAX,  "Введите
значение выводимого элемента: ");
        std::cout << "Элемент, со значением " << value << " в
списке находится "
                     "на позициях: ";
        listTimeBegin=std::chrono::system_clock::now();
        list *List=ListAndArray->List;
        for (int i=1; List!=NULL; i++, List=List->next)
          if (List->value==value)
            std::cout << i << " ";
        listTimeEnd=std::chrono::system_clock::now();
        std::cout << "\nЭлемент, со значением " << value << " в
массиве находит"
                     "ся на позициях: ";
        arrayTimeBegin=std::chrono::system_clock::now();
        for (int i=0; i<ListAndArray->listSize; i++)
          if (*(ListAndArray->Array+i)==value)
            std::cout << i+1 << " ";
        arrayTimeEnd=std::chrono::system_clock::now();
      }
```

```
    std::cout << "\nНа осуществление операции над списком было
затрачено " <<

std::chrono::duration_cast<std::chrono::nanoseconds>(listTimeEnd-
listTimeBegin).count()
                << " наносекунд.\nНа осуществление операции над
массивом было"
                "            затрачено            "            <<
std::chrono::duration_cast<std::chrono::nanoseconds>(arrayTimeEnd-
arrayTimeBegin).count()
                << " наносекунд.";
    std::cout << "\nДля возврата в меню нажмите \"Enter\".";
    std::cin.get();
  }
```

ПРИЛОЖЕНИЕ Б
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестирование программы осуществлялось на компьютере с процессором x64 под управлением ОС Windows 10 Pro (версия 20H2, 64-бит) с использованием следующих компиляторов:

- g++ (GCC) 10.2.0
- Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.27.29112 для x64

А также на компьютере с процессором x64 под управлением ОС Linux Mint 20.1 XFCE (версия ядра 5.4.0-65-generic, x86_64) с использованием следующих компиляторов:

- g++ (GCC) 9.3.0

В ходе тестирования были получены идентичные результаты независимо от использованного компилятора и ОС.

1. Тестирование чтения данных из файла:

Содержимое файла.	**Результат работы программы**
123 324 58746 435	На осуществление операции над списком было затрачено 100 наносекунд. На осуществление операции над массивом было затрачено 200 наносекунд. Текущее состояние массива: 1195 3205 587425 435 Текущее состояние списка: 1195 3205 587425 435

2. Тестирование чтения данных из терминала:

Входной поток.	**Результат работы программы**
1 2 3 4 5 6 7 8 9 10\n	На осуществление операции над списком было затрачено 100 наносекунд. На осуществление операции над массивом было затрачено 100 наносекунд. Текущее состояние массива: 1 2 3 4 5 6 7 8 9 10 Текущее состояние списка: 1 2 3 4 5 6 7 8 9 10

3. Тестирование генерации массива, заполненного псевдослучайными числами:

Входной поток.	**Результат работы программы**
20	На осуществление операции над списком было затрачено 200 наносекунд. На осуществление операции над массивом было затрачено 200 наносекунд. Текущее состояние массива: 6 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68 Текущее состояние списка: 6 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68

4. Тестирование вставки элемента в массив/список:

Входные данные	Результат работы программы
Начальное состояние массива/списка: 6 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68 Добавляемый элемент: 2 Позиция добавляемого элемента: 2	На осуществление операции над списком было затрачено 2200 наносекунд. На осуществление операции над массивом было затрачено 500 наносекунд. Текущее состояние массива: 6 2 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68 Текущее состояние списка: 6 2 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68

5. Тестирование удаления элемента из массива/списка по значению:

Входные данные	Результат работы программы
Начальное состояние массива/списка: 6 2 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68 Удаляемый элемент: 0	На осуществление операции над списком было затрачено 2000 наносекунд. На осуществление операции над массивом было затрачено 700 наносекунд. Текущее состояние массива: 6 2 45 85 29 20 96 45 85 8 57 48 90 7 12 11 67 57 24 68 Текущее состояние списка: 6 2 45 85 29 20 96 45 85 8 57 48 90 7 12 11 67 57 24 68

6. Тестирование удаления элемента из массива/списка по позиции:

Входные данные	Результат работы программы
Начальное состояние массива/списка: 6 2 45 85 29 20 96 45 85 0 8 57 48 90 7 12 11 67 57 24 68 Позиция удаляемого элемента: 10	На осуществление операции над списком было затрачено 2100 наносекунд. На осуществление операции над массивом было затрачено 600 наносекунд. Текущее состояние массива: 6 2 45 85 29 20 96 45 85 57 48 90 7 12 11 67 57 24 68 Текущее состояние списка: 6 2 45 85 29 20 96 45 85 57 48 90 7 12 11 67 57 24 68

7. Тестирование вывода значения элемента из массива/списка по позиции:

Входные данные	Результат работы программы
Начальное состояние массива/списка: 6 2 45 85 29 20 96 45 85 57 48 90 7 12 11 67 57 24 68 Позиция элемента: 3	Элемент на позиции 3 в списке: 45 Элемент на позиции 3 в массиве: 45 На осуществление операции над списком было затрачено 200 наносекунд. На осуществление операции над массивом было затрачено 100 наносекунд.

8. Тестирование вывода позиции элемента из массива/списка по значению:

Входные данные	Результат работы программы
Начальное состояние массива/списка: 6 2 45 85 29 20 96 45 85 57 48 90 7 12 11 67 57 24 68 Позиция элемента: 57	Элемент, со значением 57 в списке находится на позициях: 10 17 Элемент, со значением 57 в массиве находится на позициях: 10 17 На осуществление операции над списком было затрачено 1000 наносекунд. На осуществление операции над массивом было затрачено 900 наносекунд.

9. Тестирование скорости удаления элемента из массива/списка по позиции для массивов/списков большой длины:

Входные данные	Результат работы программы
Начальное состояние массива/списка: 13 37 58 8 82 71 85 55 5 4 72 14 5 70 66 5 63 76 18 51 97 71 15 10 24 56 71 8 48 64 82 1 53 50 60 79 62 93 10 78 32 57 62 0 15 89 2 43 70 29 50 21 39 90 25 75 24 96 5 93 15 48 88 11 35 32 72 58 76 68 82 43 92 28 48 43 71 3 9 41 57 53 42 61 59 47 42 32 89 21 97 5 31 15 28 14 67 67 28 43 31 86 35 97 35 33 66 75 39 67 84 49 81 44 69 3 31 59 91 32 82 83 57 36 10 36 36 79 46 7 26 1 51 34 88 99 7 53 66 79 0 68 98 27 17 45 49 32 47 13 18 89 4 88 18 20 34 74 95 60 55 58 68 86 93 27 28 91	На осуществление операции над списком было затрачено 1200 наносекунд. На осуществление операции над массивом было затрачено 1600 наносекунд.

91 92 0 28 12 93 17 50 17 52 82 17 40 49 74 97 69 2 41 28 90 2 72 88 12 85 48 6 10 2 79 79 86 53 85 30 1 66 1 88 80 38 0 14 5 57 17 39 56 41 9 6 83 47 95 4 8 96 64 52 79 64 29 90 48 98 34 93 43 62 47 46 27 30 63 51 63 46 69 67 20 11 68 43 94 88 40 99 87 44 4 57 7 14 18 37 70 38 67 17 46 51 19 25 83 74 65 80 18 56 73 25 89 80 84 46 31 96 8 57 32 43 37 62 8 97 68 94 58 16 55 57 32 26 75 16 85 73 69 63 65 7 41 18 60 93 69 42 88 66 77 12 2 45 84 89 19 77 98 96 79 7 2 89 45 89 75 26 45 37 92 42 80 51 87 53 15 55 82 57 60 52 67 92 26 42 41 69 16 82 65 80 96 47 67 66 91 98 26 89 17 59 9 56 80 84 21 89 60 99 93 32 49 58 88 17 96 5 60 70 20 96 41 68 76 22 85 22 70 10 7 21 Позиция элемента: 2	