**По каждой программе оформить отчет в word : математическая модель, схема алгоритма решения, текст программы.**

**Программы должны быть написаны на языке С \ С++ MS Visual Studio 2010**

## Задача 1

## Динамические массивы

***Цель лабораторной работы:*** *изучение структурной организации динамических массивов и способов доступа к их элементам с использованием указателей; совершенствование навыков процедурного программирования на языке C/С++ при решении задач обработки динамических массивов.*

***Задание на программирование:*** *используя технологию процедурного программирования, разработать программу обработки одномерных и двумерных (матриц) динамических массивов в соответствии с индивидуальным заданием.*

***Порядок выполнения работы:***

2) Разработать математическую модель: описать с помощью формул и рисунков структуру массивов и процесс их преобразования.

3) Построить схему алгоритма решения задачи.

4) Составить программу на языке C/С++.

5) Входные данные на этапах тестирования и демонстрации работы должны задаваться либо с использованием специально подобранных арифметических формул, либо вводиться с клавиатуры по запросу. **Датчики псевдослучайных чисел использовать запрещается**.

Выходные данные должны выводиться на экран с пояснениями.

6) Проверить и продемонстрировать работу программы на полном наборе тестов, в том числе с ошибочными входными данными. Входные и выходные массивы должны выводиться в одном и том же формате.

7) Использовать стандартные потоковые объекты ввода/вывода ***cin*** и ***cout***.

**8) Оформить отчет о лабораторной работе в составе: постановка задачи, математическая модель, схема алгоритма решения, текст программы, контрольные примеры.**

***Вариант***

 **1** **2**

 **8 3**

 **7 4**

 **6 5**

**9.**

1) Дан массив *a*0, *a*1, *a*2,…, *an-1*. Определить количество и произведение значений элементов массива, лежащих между элементами с максимальным по абсолютной величине значением и с минимальным по абсолютной величине значением.

2) В заданной квадратной матрице размера 2*n\**2*n* поменять местами значения элементов строк с одинаковыми номерами областей 1 и 2 (см. рисунок).

***Пример программы на обработку одномерного массива***

//Массивы динамические

//Найти и вывести номер элемента введенного с клавиатуры массива целых чисел,

//для которого сумма разностей значений с соседними элементами максимальна.

//Для крайних элементов использовать циклическое замыкание.

#include<iostream.h>

#include<math.h>

#include <locale.h>

//using namespace std ;

typedef int telem ; //объявление типа элемента массива

typedef telem \*tmas ; //объявление типа "указатель на telem"

int nomer(tmas a, telem &max, int n) ;

void inputmas(tmas a, int n) ;

int main()

{int i, n ;

 setlocale(LC\_ALL,"Russian") ;

 cout << "\n В массиве целых чисел найти номер элемента, для которого" << endl ;

 cout << " сумма разностей с соседними элементами максимальна" << endl ;

//Ввод исходных данных

 cout << "\nВведите количество элементов массива: " ;

 cin >> n ;

 tmas a = new telem [n] ; //создание переменной-указателя на telem,

 //выделение динамической памяти под массив

 //адрес начала области заносится в a

 int nom ; //номер искомого элемента

 telem max ; //значение максимальной разности

 inputmas(a, n) ;

//Поиск номера элемента

 nom = nomer(a, max, n) ;

 cout << " Искомый номер элемента массива: " << nom << endl ;

 cout << " Значение элемента:" << \*(a+nom) << " , сумма разностей=" << max ;

 delete [] a; //освобождение динамической памяти

 return 0 ;

}

int nomer(tmas a, telem &max, int n)

{telem pr ; //текущее значение разности

 int imax = 0 ; //за максимум принимаем первый по счету элемент

 max = fabs(\*(a + n - 1) - \*a) + fabs(\*(a + 1) - \*a) ;

 for(int i = 1 ; i < n - 1 ; i++)

 if(max < (pr = fabs(\*(a+i-1) - \*(a + i))+fabs(\*(a + i + 1) - \*(a+i))))

 {imax = i ;

 max = pr ;

 }

 if(max < (pr = fabs(\*a - \*(a+n-1)) + fabs(\*(a+n-2) - \*(a+n-1))))

 {imax = n - 1 ;

 max = pr ;

 }

 return imax ;

}

void inputmas(tmas a, int n)

{cout << endl <<" Введите в одной строке элементы массива, состоящего из";

 cout << endl << " " << n << " целых чисел, и нажмите <Enter>" << endl;

 for(int i = 0 ; i < n ; i++)

 cin >> \*(a +i) ;

}

***Пример программы на обработку двумерного массива (матрицы)***

//Массивы динамические двумерные

//Программа находит строку введенного с клавиатуры

//двумерного массива целых чисел, содержащую максимальную

//сумму значений элементов

#include<iostream.h>

#include <locale.h>

//using namespace std;

typedef int telem ; //определение типа элементов массива

typedef telem \*tstr ; //определение типа "указатель на telem"

typedef tstr \*tmatr; //определение типа "указатель на указатель на telem"

void inputmatr(tmatr a, int str, int sto) ;

void outputmatr(tmatr a, int str, int sto) ;

telem poisk\_str(tmatr a, int str, int sto, int &imax) ;

int main()

{ int i, j ;

 int imax; //номер строки с максимальной суммой значений элементов

 int n ; //число строк матрицы

 int m ; //число столбцов матрицы

 tmatr a ; //объявляется переменная-указатель на указатель на telem

 telem max ; //максимальная сумма значений элементов

 setlocale(LC\_ALL,"Russian") ;

 cout << "\nВведите количество строк и столбцов матрицы: " ;

 cin >> n >> m;

 a = new tstr [n] ; //выделение динамической памяти под массив

 //указателей на строки массива

 for(i = 0; i < n; i++) //цикл выделения памяти под каждую строку

 \*(a + i) = new telem [m] ; //каждому элементу массива указателей

 //на строки присваивается адрес начала

 //области памяти, выделяемой под строку

 inputmatr(a, n, m) ;

 cout << "Исходная матрица:\n" ;

 outputmatr(a, n, m) ;

 max = poisk\_str(a, n, m, imax) ;

 cout << "Максимальная сумма=" << max << " элементов строки содержится" ;

 cout << endl << "в " << imax << "-й строке исходного массива " << endl ;

 cout << endl << "Для завершения нажмите <Enter>" ;

//Освобождение динамической памяти

 for(i = 0; i < n; i++)

 delete a[i] ;

 delete a ;

 return 0 ;

}

void inputmatr(tmatr a, int str, int sto)

{ cout << "Введите построчно через пробел элементы" << endl ;

 cout << "двумерного массива размера " << str << "x" << sto << endl ;

 cout << "После ввода строки нажимайте <Enter>" << endl ;

 for(int i = 0 ; i < str ; i++)

 for(int j = 0 ; j < sto ; j++)

 cin >> \*(\*(a + i) + j) ;

}

telem poisk\_str(tmatr a, int str, int sto, int &imax)

{telem max, //максимальная сумма значений элементов

 pr ; //текущая сумма

 imax = 0 ; //предполагаем, что максимальная сумма значений в 0-ой строке

 max = 0 ;

 for(int j = 0 ; j < sto ; j++)

 max += \*(\*(a + 0) + j) ;

 for(int i = 1 ; i < str ; i++)

 {pr = 0 ;

 for(j = 0 ; j < sto ; j++)

 pr += \*(\*(a + i) + j) ;

 if(max < pr)

 {imax = i ;

 max = pr ;

 }

 }

 return max ;

}

void outputmatr(tmatr a, int str, int sto)

{ int i, j ;

 for(i = 0; i < str; i++)

 {for(j = 0; j < sto; j++)

 cout << \*(\*(a + i) + j) << ' ' ;

 cout << '\n';

 }

}

## Задача 2

## Линейные списки

***Цель лабораторной работы:*** *изучение способов создания и принципов использования односвязных линейных списков; изучение стандартных средств языка C/C++ для работы с динамической памятью; совершенствование навыков модульного программирования на языке C/С++ при решении задач обработки линейных списков; изучение способов разработки многофайловых проектов.*

***Задание на программирование:*** *используя технологию процедурного программирования разработать программу обработки односвязных линейных списков с числом элементов в списке* *не менее десяти в соответствии с индивидуальным заданием.*

***Порядок выполнения работы:***

2) Построить схему алгоритма решения задачи с использованием функций создания, просмотра, обработки списка, удаления списка из динамической памяти.

3) Составить спецификации функций.

4) Составить программу на языке C/С++.

5) Проверить и продемонстрировать работу программы на полном наборе тестов. Обеспечить одновременный показ на экране исходного и результирующего списков.

**6) Оформить отчет о лабораторной работе в составе: постановка задачи, математическая модель, схема алгоритма решения, спецификации функций, текст программы, контрольные примеры.**

***Вариант***

**9.**

Удалить из списка L за каждым вхождением элемента со значением Е один элемент, если он есть и его значение отличено от Е.

***Пример программы***

//Создать линейный список с символьной информационной частью

//Преобразовать линейный список в циклический, взяв в качестве

//головного элемента элемент с № p

//Многофайловый проект

//

**//Основной файл spis.cpp**

#include<iostream.h>

#include<locale.h>

#include"modulSpi.h"

//основная программа

int main()

{int i = 1, p;

 list \*first = NULL;

 char ch;

 setlocale(LC\_ALL,"Russian") ;

 cout << "\n Вводите элементы списка сплошной строкой;"

 "\n в конце - точка:";

 cout << "\n ";

 cin >> ch;

 while(ch != '.')

 {first = sozdspis(ch, i, first);

 i++;

 cin >> ch;

 }

 cout << "Введите номер первого элемента для циклического списка\n";

 cin >> p;

 first = zicl(p,first);

 if(first != NULL)

 {cout << "Результат получен\n";

 viewsp(first);

 }

 else cout << "Результата нет, элемента с таким номером в списке нет\n";

 cin.get();

 return 1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//

**//Заголовочный файл**

//Помещаем сюда определения новых типов и объявления функций

//Файл modulSpi.h

//Объявления типов

struct list

{char data ; //поле данных

 int numb ; //поле номера элемента

 list \*link ; //поле адреса

} ;

//Объявления (прототипы) функций

list \*sozdspis(char, int, list\*) ;

list \*zicl(int, list\*) ;

void viewsp(list\*) ;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**//Исходный файл**

//Реализация набора функций для обработки списка

//Файл modulSpi.cpp

//

#include <iostream.h>

#include "modulSpi.h"

//zicl Begin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Зацикливание списка

list \*zicl(int q, list \*first)

{list \*now = first, //указатель now устанавливаем на голову списка

 \*result;

 while((now->numb != q) && (now != NULL)) now = now->link;

 result = now; //result содержит указатель на элемент с номером q

 //или NULL, если такого элемента нет

 now = first; //указатель устанавливаем на голову списка

 while(now->link != NULL) now = now->link; //сдвигаем указатель на конец

 now->link = first; //зацикливаем

 return result;

}

//zicl End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//viewsp Begin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Просмотр циклического списка

void viewsp(list \*head)

{list \*tec = head;

 do

 {cout << tec->data;

 tec = tec->link;

 }

 while(tec != head);

}

//viewsp End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//sozdspis Begin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Добавление нового элемента в список

list \*sozdspis(char ch, int k, list \*head)

{list \*tec,

 \*nov = new(list);

 nov->data = ch;

 nov->numb = k;

 nov->link = NULL;

 if (head) //список не пуст

 {tec = head;

 while (tec->link)

 tec = tec->link;

 tec->link = nov;

 }

 else //список пуст

 head = nov;

 return head;

}

//sozdspis End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_