**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В. И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

### Тема: **ДВУМЕРНЫЕ СТАТИЧЕСКИЕ МАССИВЫ. УКАЗАТЕЛИ**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. |  | Коврижных А.Н. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цели работы:** разработать алгоритм и написать программу, которая заполняет, изменяет и сортирует матрицу с помощью указателей.

**Основные теоретические положения:**

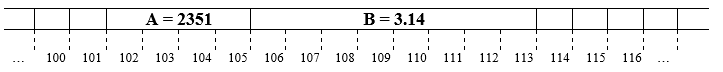
Указатели и ссылки являются одними из самых важных и достаточно сложных для понимания и использования средств языка программирования. Они ориентированы на прямую работу с памятью компьютера. С помощью этих средств реализуется работа с динамической памятью и динамическими объектами, возвращение из функций измененных данных и многое другое. К использованию указателей и ссылок мы будем неоднократно возвращаться в последующих разделах.

Все данные (переменные, константы и др.) хранятся в памяти. Память представляет собой непрерывную последовательность ячеек (байтов), каждая из которых имеет свой номер – адрес:

**Изображение выглядит как линия, снимок экрана, текст, Шрифт

Автоматически созданное описание**

При определении, например, некоторой переменной, она располагается в памяти по определенному адресу и занимает столько ячеек, сколько требует тип этой переменной. Пусть, например, имеется переменные **int A = 2351** и **double B = 3.1** и пусть они располагаются в памяти так:



Говорят, что переменная **А** располагается по адресу 102 и занимает 4 байта, а переменная **B** имеет адрес 106 и занимает 8 байт памяти.

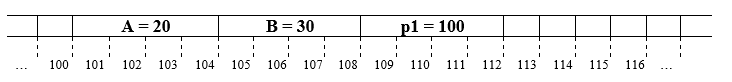
К указателям можно применять некоторые арифметические операции. К таким операциям относятся:  **+**,**-**, **++**, **--**. Результаты выполнения этих операций по отношению к указателям существенно отличаются от результатов соответствующих арифметических операций, выполняющихся с обычными числовыми данными.

Рассмотрим следующий пример:

int A = 20, B = 30;

int \*p1 = &A;

Пусть переменные **A**и **B** расположены в памяти, например, так, как это показано на следующем рисунке:



Указатель **p1** содержит адрес переменной **A**, который равен 100 и \***p1** будет равно значению переменной **A**, то есть 20. Выполним следующую операцию:

p1 = p1 + 1;

или, что то же самое:

p1++;

Значение указателя изменится и станет равным 104, а не 101, как, наверное, ожидалось. То есть теперь указатель ссылается уже на переменную **B**и значение \***p1**будет равно 30.

Таким образом, добавление или вычитание 1 из указателя приводит к изменению его значения на размер базового типа указателя. В общем случае, например, при выполнении следующей операции:

p1 = p1 + N;  //  N – некоторое целое значение

значение указателя увеличится на **sizeof(<базовый тип указателя>) \* N** и в нашем случае это приращение будет равно **sizeof(int) \* N = 4 \* N**. Так, если N = 4, а p1= 100, то значение указателя **p1** увеличится на 16 и станет равно 116, и указатель будет  ссылаться на данные, расположенные по адресу 116.

Когда мы определяем в программе некоторый массив, например,

int Arr[10];

переменная **Arr** без индексов представляет собой указатель на первый элемент массива в данном случае из 10 целых чисел (содержит адрес первого элемента массива). Если вывести на экран значение переменной **Arr**

cout << Arr:

мы увидим некоторое целое значение в шестнадцатеричном формате, соответствующее адресу первого элемента этого массива.

ода мы определяем в программе некоторый массив, например,

int Arr[10];

переменная **Arr** без индексов представляет собой указатель на первый элемент массива в данном случае из 10 целых чисел (содержит адрес первого элемента массива). Если вывести на экран значение переменной **Arr**

cout << Arr:

мы увидим некоторое целое значение в шестнадцатеричном формате, соответствующее адресу первого элемента этого массива.

**Постановка задачи:**

1) Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка N (6,8,10) случайными числами от 1 до  N\*N согласно схемам, приведенным на рисунках. Пользователь должен видеть процесс заполнения квадратичной матрицы (\*Для манипуляции с элементами используйте только арифметику указателей):

Изображение выглядит как Прямоугольник, зарисовка, диаграмма, домкрат

Автоматически созданное описание

2) Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами (\*Для манипуляции с элементами используйте только арифметику указателей):

**Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание**

3)Используя арифметику указателей, сортирует элементы любой сортировкой из списка ниже (если во 2 ПР вы реализовывали одну из представленных сортировок, рекомендуется переиспользовать написанный код и модернизировать его для работы с указатями).   
Варианты сортировок:

* Shaker sort;
* Comb sort;
* Insert sort;
* Quick sort;

4)Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число (\*Для манипуляции с элементами используйте только арифметику указателей).

**Выполнение работы:**

В функции creatMatrix() создается матрица размерности N.

int\* matrix = new int[N \* N]

Оператор *new* выделяет место в оперативной памяти под матрицу. После *new* всегда должно быть прописано *delete[]* оператор, что удаляет место под матрицу.

При запуски программы пользователь может выбрать размерность матрицы.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рис. 1. – Размер матрицы.

После выбора размерности матрицы, пользователь выбирает способы заполнения матрицы.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 2. –Заполнение матрицы.

Также пользователь может пересобрать матрицу, а именно изменить размер матрицы.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис. 3. – Заполненная матрица.

После заполнения матрицы пользователь выбирает, как хочет изменить матрица. Также он может вернуться к предыдущим пунктам.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 3. – Изменённая матрица.

После изменение матрицу нужно нажать на любую клавишу, чтобы перейти на следующий пункт.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рис. 5 – Сортировка матрицы.

После сортировки матрицы. Пользователь может вернуться к предыдущим пунктам задание. В работе используется быстрая сортировка.

**Вывод:**

В ходе лабораторной работе и больше узнал об указателях и об матрицы. Также научился обращаться к элементам матрицы по указателям, а не по индексам.