

Центральноукраїнський національний технічний університет
Кафедра програмування комп'ютерних систем і мереж
Дисципліна: Базові методології та технології програмування

Звіт

з лабораторних робіт №14-18

Теми: Вивчення технологій розв'язку задач з використанням циклів,
одновимірних, двовимірних, динамічних масивів та вивчення технологій
обробки текстових рядів

Мета: Ознайомитись з операторами циклів, принципами формування
одновимірних, двовимірних, динамічних масивів, вивчити способи
виділення пам'яті для динамічних масивів, навчитись будувати
математичну модель задачі, вивчити методи перевірки правильності
вхідних даних, вивчити можливості оголошення та ініціалізації рядів,
вивчити головні функції роботи з рядами, засвоїти використання
головних функцій бібліотек та отримати практичні навички в написанні
програм.

Виконав: ст. гр. КМ-20

Калиновський В.М.

Перевірила: доцент кафедри ПКСМ

Рибакова Л.В.

Варіант 3

Дата виконання: 28 березня 2021 — 30 березня 2021

Дата оформлення: 31 березня 2021

Кропивницький 2020 р.

Завдання 1. Написати програму, що формує послідовність з парних A_n , кратних тринадцяти з діапазоном від 0 до 300, рахує суму цих самих елементів і виводить результат.

Завдання 2. Написати програму, що шукає всі ПРОСТІ числа в діапазоні від 0 до 300 і виводить результат.

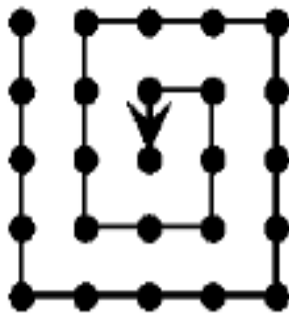
Завдання 3. Написати програму, що шукає і виводить результат перших ненульових коренів a , b і c в рівнянні $(a * b) - c^3 - 9 = 0$.

Завдання 4. Написати програму, в якій за вибором користувача (вручну або програмно) заповнюється масив $Mass[15]$ і підраховується кількість парних елементів масиву (результат виводиться).

Завдання 5. Написати програму, що виконує запис значень елементів двовимірного масиву $Mass[5][5]$ у послідовності, зазначеній на *Ілюстрації 1*, вивести результати читання і запису, значення матриці повинні бути в діапазоні від 1 до 25.

Завдання 6. Написати програму для заповнення двовимірного масиву $Mass[5][5]$ значеннями "0" і "1" у послідовності, зазначеній на *Ілюстрації 2* та вивести результат.

Ілюстрація 1: Послідовність до завдання 5.



Ілюстрація 2: Схема до завдання 6.

1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1

Завдання 7. Написати програму, котра має:

1. ДИНАМІЧНІ масиви;
2. У масиву максимальний розмір - $N * M$, де N і M - статичні константи;
3. У масиву реальний розмір - $n * m$ ($n < N$, $m < M$) - вводяться користувачем;
4. Інтервал генерації елементів, який вказує користувач;
5. У масиву рандомно генеруються елементи;
6. На вході дані і елементи вхідного і вихідного масивів;
7. Висновок результату в красивому вигляді;
8. Функцію - зміна значень елементів масиву на значення відповідних елементів заданого одновимірного масиву.

Завдання 8. Потрібно, щоб користувач вводив рядок тексту і програма виконувала наступне:

1. Вважає кількість різних слів, що є в заданому тексті
2. Виводить на екран кількість використаних символів
3. Видаляє всі слова, в яких є сдвденние літери

Робота. Я вирішив усі завдання написати в одній програмі. Кожен код програми буде представляти свій клас, а у головній функції будуть міститися тільки конструктори об'єктів й інколи виклики деяких методів класа. Кожен клас обрамлено своїми препроцесорами `#ifdef` та `#endif`, а на початку прописано `#define`. В залежності від того, що прописано у `#define`, компілятор буде вибирати тільки той блок коду, який обрамлено `#ifdef` с такою ж назвою, та весь інший код, який не обрамлено іншими `#ifdef`. Напочатку опишемо, що можна прописати у `#define`, і що від цього буде скомпільовано. Після цього можна приступати до написання завдань.

! Зауваження: всі схеми та алгоритми я буду описувати псевдокодом, а не блок-схемами, так як його швидше робити в декілька десятків разів, він менше місця займає і його набагато легше зрозуміти. Якщо потребується — я додам блок-схеми окремо, написаними від руки.

Для завдання 1 розроблено наступну загальну структуру програми:

Крок 1: Перебирати елементи від 0 до 300;

Крок 2: Перевіряти кожен елемент: чи ділиться елемент на 2;

Крок 3: Перевіряти елементи, що ділиться на 2 на те, чи ділиться вони ще на 13;

Крок 4: Якщо так, то виводити їх на екран;

Крок 5: Додавати ці елементи до суми;

Крок 6: Коли дораховано до 300 — вивести суму;

Крок 7: Після виведення суми завершити програму.

Що очікується від програми? Вона повинна відбирати тільки ті числа, що діляться на тринадцять (13, 26, 39, 52 ...), але не всі, а тільки ті, що ще є парними (26, 52, 78 ...), тобто першим має бути 26 і кожне наступне буде більше попереднього на 26. Напишемо програму, що вибератиме всі парні числа, що діляться на 13. Результат зображено на ілюстрації 3.

Ілюстрація 3: Результат виконання першої програми.

```

Y:\Работы\Лаб\БМТП\Семестр2\1_DefineONE.exe
Подготовил Калиновский Валентин.
Первое задание (лабораторная №1):
26 + 52 + 78 + 104 + 130 + 156 + 182 + 208 + 234 + 260 + 286 = 1716
Конец задания.

```

Результат: як бачимо, перші три числа співпали. Я власноруч перевіряв усі ці числа й вони реально діляться на 13 (що вони парні й так видно). Якщо перевіряти цих суму, елементів — вона дійсно має бути такою. *Вважаємо завдання виконаним!* ✓

Для завдання 2 написана загальна структура програми так:

Крок 1: Перебирати кожне число від 1 до 300;

Крок 2: Перевіряти кожне таке число, чи ділиться воно на ще якесь від 2 до 300;

Крок 3: Якщо ділиться, то скинути лічильник і почати з наступного числа;

Крок 4: Якщо інакше — продовжувати операцію далі;

Крок 5: Якщо перерахунок закінчився й число ніколи не поділилося — воно є простим й ділиться тільки на 1 й на себе;

Крок 6: Виводжу просте число;

Крок 7: Повторюю кроки 2 - 6 з наступними числами до 300;

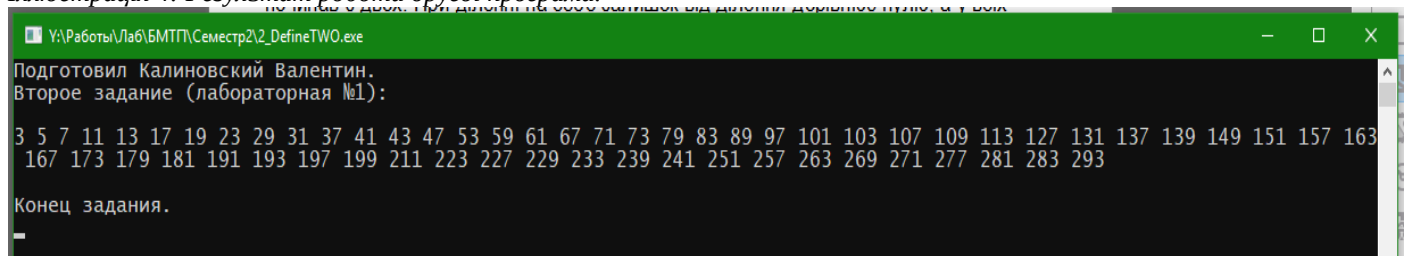
Крок 8: Коли дораховано до 300 — завершити програму.

! Пояснення: як саме зрозуміти на кроці 5, що число ділиться тільки на 1 і на саме себе? Відкрию секрет, але на один діляться всі числа, які існують (крім нуля, звісно ж), тому я його навіть не використовував у розрахунках для оптимізації програми, а починав з двох. При діленні на себе залишок від ділення дорівнює нулю, а у всіх інших випадках має залишатись тільки залишок від ділення — це я використовував при написанні програми. Якщо ж буде ще якесь число, на яке ділиться вибране націло, то залишок теж буде дорівнювати нулю — у цьому випадку виконується інструкція кроку 3. І якщо не виконається жодного разу інструкція за кроком три, це означатиме, що програма поділила вибране число націло тільки при діленні на себе. Це число ми і виводимо. Так перебираються всі числа.

Можна код класу програми переписати в окремий код, прописати `#define LIMIT` (замість підкреслень вписати обмеження) й у циклі `while` прописати замість “300” - “LIMIT”. У цьому випадку програма буде рахувати прості числа не до 300 а до будь-якого числа, яке прописано у `#define`, хоч до трильйона (але не радиться це робити, оскільки програма це може робити довго — чим більше число, тим більше потрібно часу для розрахунку). Або просто спитати у користувача, до якого числа порахувати прості числа, записати відповідь у змінну й замінити у циклі 300 або LIMIT на цю змінну. Тоді програма буде рахувати до тих чисел, які задає користувач.

Що очікується від програми? Потрібно отримати прості числа до 300 типу 1, 2, 3, 5, 7... Результат зображено на ілюстрації 4.

Ілюстрація 4: Результат роботи другої програми.



! Консоль у мене розширена, при зменшенні строки не перевиводяться, а при зменшенні і перевідкритті консоль знову стає широку. Скріншот я можу додати тільки такий.

Результат: я власноруч перепровірив усі ці числа з таблицею простих чисел. Вони дійсно є всі у тій таблиці, а також на вивід не потрапляють ніякі зайві числа й жодного загубленого. **Вважаємо завдання виконаним! ✓**

Для завдання 3 написано таку загальну структуру програми:

Крок 1: Нехай найближчими до нуля будуть варіанти з числами до 10;

Крок 2: Перебираємо всі числа до 10:

Крок 2.1: Перебираємо число а до десяти;

Крок 2.2: Перебираємо число b до десяти;

Крок 2.3: Перебираємо число с до десяти;

Крок 2.4: Перебираємо число а з b до десяти;

Крок 2.5: Перебираємо число а з с до десяти;

Крок 2.6: Перебираємо число b з с до десяти;

Крок 2.7: Перебираємо число а з b з с до десяти;

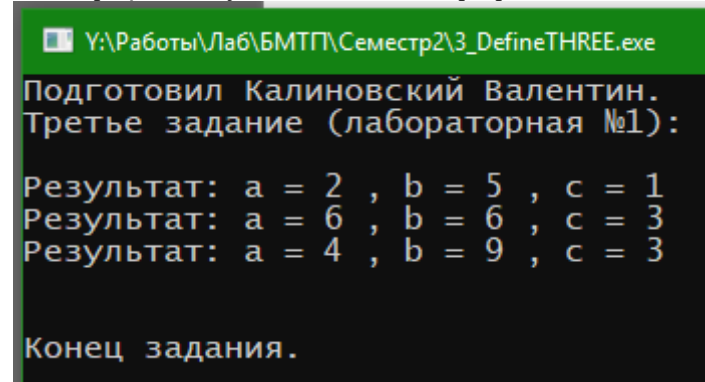
Крок 3: Кожне з вибраних чисел перевіряємо по формулі $(a*b)-c^3-9=0$;

Крок 4: Якщо повертається true — вивести результат;

Крок 5: Після перебору всіх варіантів — завершити програму.

Що очікується від програми? Насправді будуть перебиратись всі можливі комбінації для трьох чисел від 1 до 10. Для першого числа 10 варіацій, для другого 10 і для третього, а загальна кількість комбінацій $10 * 10 * 10 = 1000$. Комп'ютер перебирає всю 1000 комбінацій і перевіряє кожну на те, чи буде ця комбінація у формулі дорівнювати 0. Результат зображено на ілюстрації 5.

Ілюстрація 5: Результат виконання програми 3.



```
Y:\Работы\Лаб\БМТП\Семестр2\3_DefineTHREE.exe
Подготовил Калиновский Валентин.
Третье задание (лабораторная №1):

Результат: a = 2 , b = 5 , c = 1
Результат: a = 6 , b = 6 , c = 3
Результат: a = 4 , b = 9 , c = 3

Конец задания.
```

Результат: отримано з 1000 комбінацій усього 3. Якщо підставити ці комбінації у формулу — вона дійсно буде дорівнювати 0. Власноруч перевіряти всі інші 997 варіацій у мене немає часу — будемо вважати, що програма рахує правильно. Щодо трьох результатів замість одного скажемо так: у кожному варіанта найменше число менше максимального іншого варіанта, тому вибрати найменший неможливо з цих трьох. Якщо вибирати по сумі елементів — перший найменший, але у нас не авторитарна система, тому у користувача має бути вибір — залишимо три варіанти. Якщо потрібно один — перевірку на суму можна написати у дві строчки й це може зробити кожний, а перевіряються уміння на знання операторів розгалужень з перших занять. *Тому будемо вважати завдання виконаним до кінця!* ✓

Для завдання 4 описано загальну структуру програми так:

Крок 1: Питаємо у користувача, як заповнити масив;

Крок 2: Якщо вибрано перший варіант — користувач власноруч вводить всі 15 елементів масива;

Крок 3: Якщо вибрано другий варіант — елементи масива генеруються рандомно;

Крок 4: Якщо відповідь введено неправильно — виводиться повідомлення про неправильну відповідь й обирається другий варіант;

Крок 5: Заповнення/генерація масива;

Крок 6: Для кожного елемента шукається пара;

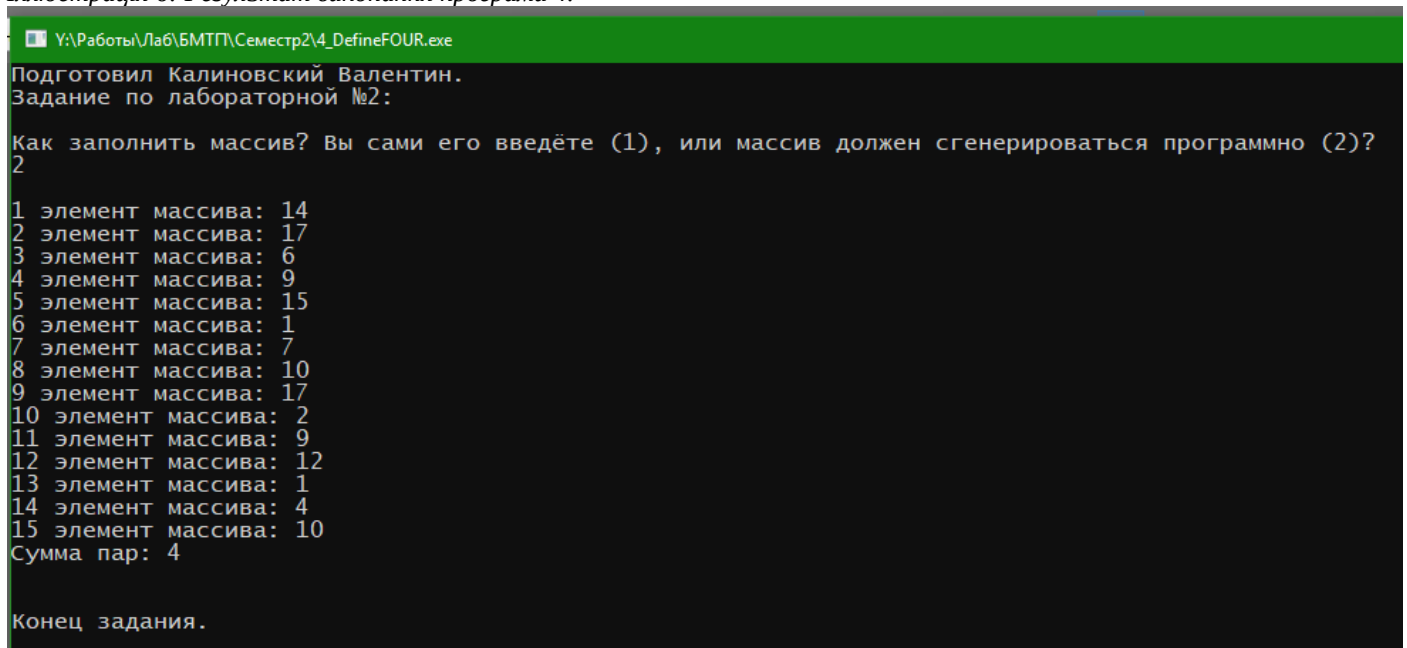
Крок 7: Рахуємо кількість пар й виводимо результат;

Крок 8: Після вивода програми програма завершається.

! Зауваження: на момент написання програми до мене ще не дійшло, що під парними мається на увазі “чётные” — я тоді переклав на російську як “парные”, тому я й написав так програму. Тільки на момент написання звіту під час опису першого завдання, коли мені треба було написати українською “чётные” й отримав у перекладачі “парні” — я зрозумів, що у іншій програмі мабуть малося на увазі інший результат. Заново переписувати програму я не буду — будемо вважати що така умова малася на увазі. Тим паче навіть копіювати код з іншої програми? Краще щось нове виконати. Тому далі буде описана програма з таким кодом, як було описано раніше. А якщо дійсно малося написати таку програму, як я зрозумів, вважайте це зауваження неактуальним.

Що очікується від програми? Користувач уводить елементи масива сам, чи вибирає автоматичну генерацію, й на виході отримує кількість парних елементів. Результат зображено на ілюстрації 6.

Ілюстрація 6: Результат виконання програми 4.



```
Y:\Работы\Лаб\БМТП\Семестр2\4_DefineFOUR.exe
Подготовил Калиновский Валентин.
Задание по лабораторной №2:

Как заполнить массив? Вы сами его введёте (1), или массив должен сгенерироваться программно (2)?
2

1 элемент массива: 14
2 элемент массива: 17
3 элемент массива: 6
4 элемент массива: 9
5 элемент массива: 15
6 элемент массива: 1
7 элемент массива: 7
8 элемент массива: 10
9 элемент массива: 17
10 элемент массива: 2
11 элемент массива: 9
12 элемент массива: 12
13 элемент массива: 1
14 элемент массива: 4
15 элемент массива: 10
Сумма пар: 4

Конец задания.
```

Результат: як бачимо, елементи дійсно генеруються і якщо ми порахуємо власноруч — отримаємо, що програма правильно рахує кількість парних елементів масива (елементів які мають пару, а не діляться на 2). *Вважаємо завдання виконаним! ✓*

Для завдання 5 написано загальну структуру програми так:

Крок 1: Оголошуємо масив 5*5;

Крок 2: Передаємо його у конструктор;

Крок 3: Генеруються елементи масива від 10 до 25;

Крок 4: Записуються ці елементи у ряд;

Крок 5: За завданням моя програма має ЗАПИСУВАТИ ці згенеровані елементи за певною послідовністю, тому я замінюю з певного елемента масива у певному напрямку на певну кількість елементів з певного ряда нульові елементи:

Крок 5.1: Перевіряю, у якому напрямку йде зміна (по x чи y), вектор зміни (збільшення чи зменшення) і наскільки треба просунутися у заданому напрямку;

Крок 5.2: Починається зміна з певної координати;

Крок 5.3: Виконується проходження, замінюючи вказану кількість елементів;

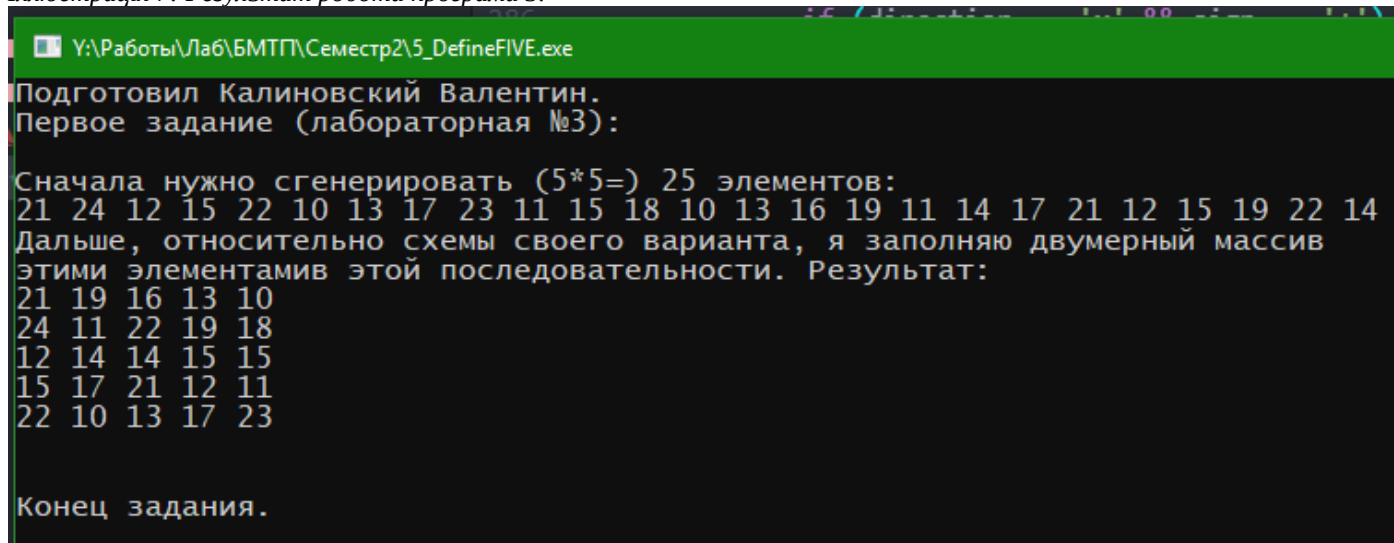
Крок 6: Після виконання всіх викликів виводжу результат сформованого масиву;

Крок 7: Після вивода результату програма завершається.

! Зауваження: за кроком 3 треба було генерувати елементи від 1 до 25, але для краси результату було вирішено обмежити цей інтервал з 1 до 10. Пояснення написано у коментарі у коді.

Що очікується від програми? Виводиться масив згенерованих у тому порядку який задано у завданнях на ілюстрації 1. Результат зображено на ілюстрації 7.

Ілюстрація 7: Результат роботи програми 5.



```
Y:\Работы\Лаб\БМТП\Семестр2\5_DefineFIVE.exe
Подготовил Калиновский Валентин.
Первое задание (лабораторная №3):

Сначала нужно сгенерировать (5*5=) 25 элементов:
21 24 12 15 22 10 13 17 23 11 15 18 10 13 16 19 11 14 17 21 12 15 19 22 14
Дальше, относительно схемы своего варианта, я заполняю двумерный массив
этими элементами этой последовательности. Результат:
21 19 16 13 10
24 11 22 19 18
12 14 14 15 15
15 17 21 12 11
22 10 13 17 23

Конец задания.
```

Результат: спочатку генерується ряд елементів, а потім він записується у масив. Це можна перевірити у коді. *Вважаємо завдання виконаним! ✓*

Для завдання 6 уявімо, що масив 5×5 є одномірним масивом із 25 елементів. Елементи між собою відносяться так: $[0][0] = [0]$, $[0][1] = [1]$, ..., $[0][4] = [4]$, $[1][0] = [5]$, $[1][1] = [6]$, ..., $[4][3] = [23]$, $[4][4] = [24]$ (нагадаємо, що нумерація починається з 0, а не з 1, тому п'ятий елемент буде під номером 4), тому опишеться загальна структура програми так:

Крок 1: Переберемо кожен з 25 елементів;

Крок 2: Кожен парний елемент буде 0, а не парний 1;

Крок 3: Запишемо кожен елемент до матриці;

Крок 4: Після перебірки виведемо готову матрицю;

Крок 5: Після виведення масива програма завершається.

Що очікується від програми? Масив наступного вигляду:

1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1

Результат зображено на ілюстрації 8.

Ілюстрація 8: Результат роботи програми 6.

Результат: Програма заповнює масив нолями та одиницями так, як потрібно.
Вважаємо завдання виконаним! ✓

Для завдання 7 необхідно написати багато коду (так кажеться на перший вигляд), але якщо не брати до уваги частину зі спілкування, то кода там написано менше, ніж у деяких інших програми цих завдань. Загальна структура програми виглядає так:

Крок 1: Дізнатися у користувача розмір масива, який йому потрібен;

Крок 2: Якщо зазначені розміри менше максимальних — дізнатися про межі генерації елементів;

Крок 3: Динамічно оголосити масив;

Крок 4: Згенерувати елементи масива;

Крок 5: Вивести масив;

Крок 6: Динамічно оголосити ряд (у завданні ніде не було задано цього ряду, тому мається на увазі, що програма має його згенерувати як масив);

Крок 7: Згенерувати елементи ряду;

Крок 8: Вивести ряд;

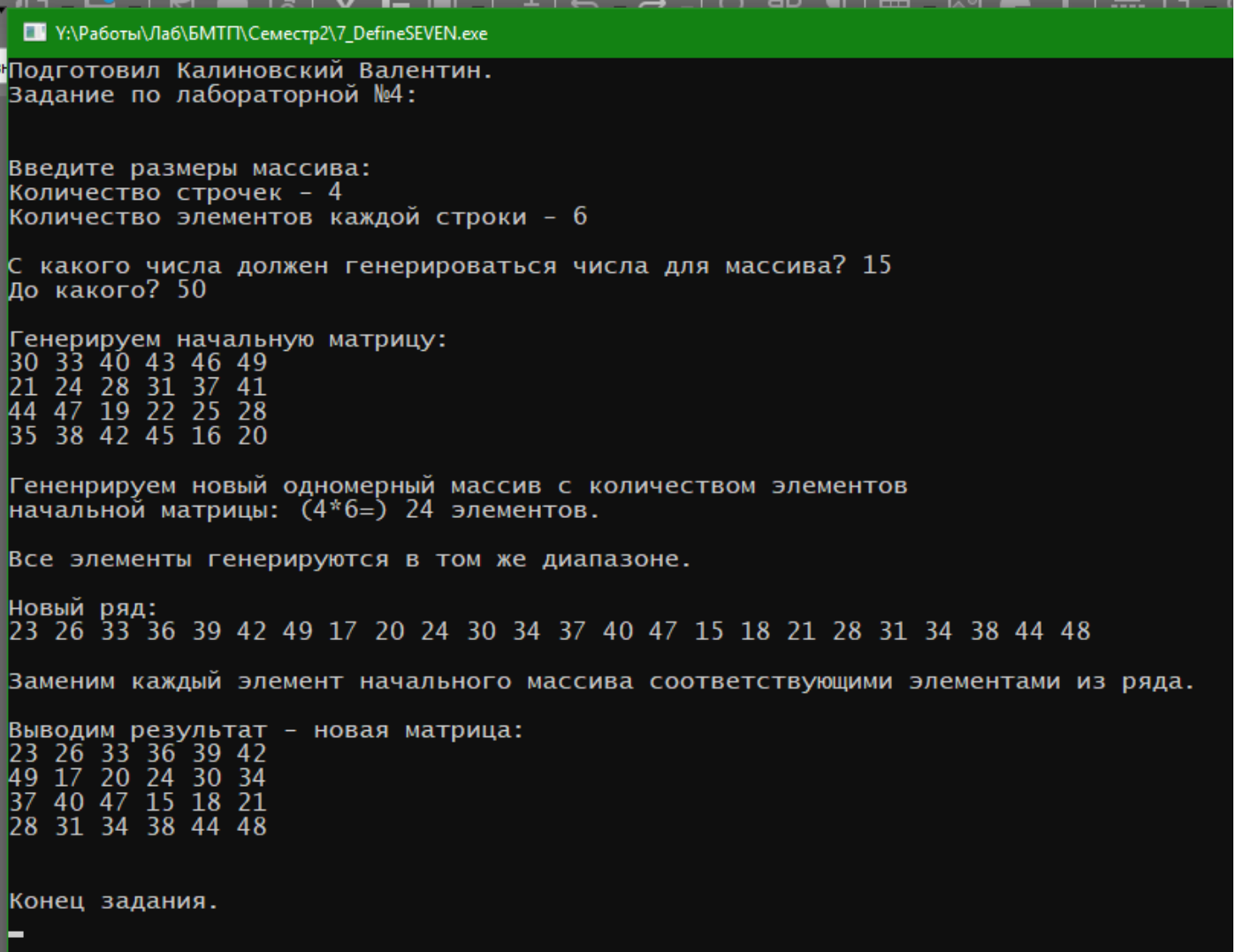
Крок 9: Замінити елементи масива відповідними елементами ряду (як це робити я розказано вже два рази у цьому звіті);

Крок 10: Вивести новий масив;

Крок 11: Після виведення масива програма завершається.

Що очікується від програми? Має згенеруватись і вивести перший масив, ряд (елементи якого мають замінити елементи масива) і новий масив (після заміни елементів). Результат зображено на ілюстрації 9.

Ілюстрація 9: Результат роботи програми 7.



```
Y:\Работы\Лаб\БМТП\Семестр2\7_DefineSEVEN.exe
Подготовил Калиновский Валентин.
Задание по лабораторной №4:

Введите размеры массива:
Количество строчек - 4
Количество элементов каждой строки - 6

С какого числа должен генерироваться числа для массива? 15
До какого? 50

Генерируем начальную матрицу:
30 33 40 43 46 49
21 24 28 31 37 41
44 47 19 22 25 28
35 38 42 45 16 20

Генерируем новый одномерный массив с количеством элементов
начальной матрицы: (4*6=) 24 элементов.

Все элементы генерируются в том же диапазоне.

Новый ряд:
23 26 33 36 39 42 49 17 20 24 30 34 37 40 47 15 18 21 28 31 34 38 44 48

Заменяем каждый элемент начального массива соответствующими элементами из ряда.

Выводим результат - новая матрица:
23 26 33 36 39 42
49 17 20 24 30 34
37 40 47 15 18 21
28 31 34 38 44 48

Конец задания.
```

Результат: Програма працює так, як задано завданням: генерує масив, ряд, і замінює елементи масива елементами з ряду. Зауважу, що пам'ять під масив і ряд виділяється динамічно. Ще відмічу, що на генерацію у цій програмі було затрачено рівно одна хвилина — 30с на масив і 30с на ряд. *Вважаємо завдання виконаним!* ✓

Для завдання 8 треба працювати вже зі строками, які, свого роду, є масивами, але зберігають не цифри, а символи. Опишемо загальну структуру програми так:

Крок 1: Програма запам'ятовує уведену користувачем строку;

Крок 2: Програма перебирає усі символи рядка;

Крок 3: Коли знайдено “пробіл” — збільшити показник кількості слів на одиницю (так як після останнього слова нема пробілу — програма не буде рахувати це слово, тому програма завжди буде стартувати вже зі значенням “1”: хоча б одна буква вже може бути словом);

Крок 4: Коли знайдено “пробіл”, а перед ним стоїть знак “тире” — не збільшувати показник кількості слів (всі знаки пунктуації, окрім “тире”, пишуться відразу після слова й тільки “тире” пишеться так, що з обох його сторін має стояти пробіл — правила пунктуації у мові. Взагалі кажучи, ця програма вважає всі знаки пунктуації частиною слова, а тому і одинокі “тире” буде рахуватись словом. Щоб такого не виникало — написана ця перевірка);

Крок 5: Виводиться кількість слів;

Крок 6: Перебрати кожен символ рядка до ‘\0’ (або “порахувати кількість введених користувачем символів”);

Крок 7: Вивести кількість символів рядка;

Крок 8: Поділити ряд на слова;

Крок 9: У кожному слові перевірити сусідні букви;

Крок 10: Якщо співпадіння нема — занести слово у новий ряд;

Крок 11: Якщо є співпадіння — пропустити слово;

Крок 12: Вивести нове речення;

Крок 13: Після виведення результату програма завершається.

Що очікується від програми? Програма запрошує ввести речення, й після отримання результату виводить кількість слів, символів і нове речення без слів, у яких є сподвоєнні літери. Результат зображено на ілюстрації 10.

Ілюстрація 10: Результат виконання програми 8.

```
Y:\Работы\Лаб\БМТП\Семестр2\8_DefineEIGHT.exe
Подготовил Калиновский Валентин.
Задание по лабораторной №5:
Введите строку: Здравствуйте! Меня зовут Валентин и я - программист. Этим всё сказано.
Количество слов в вашем тексте: 10
Количество написанных символов: 70
Новое предложение без слов с удвоенными буквами:
"Здравствуйте! Меня зовут Валентин и я - Этим всё сказано."
Конец задания.
```

Результат: Я власноруч перерахував букви й слова — все співпадає. Нове

речення теж правильно написано — подвоєнні літери є тільки в слові програміст (всі програми розраховані під російську мову). *Вважаємо завдання виконаним! ✓*

*! Зауваження про генерацію масивів: для рандомної генерації за ядро генератора береться поточний час, а так як генерація відбувається дуже швидко — за ядро береться один час і генерується одне й те саме число. Щоб цього уникнути — потрібен інтервал >1 секунди, тому генерація кожного елемента завжди відбуватиметься 1,25 секунди. Умовно кажечи, масив 15*15 буде генеруватися ($15 * 15 = 225$ елементов * $1,25 = 281,25$ секунди / 60 =) трішки більше 4,5 хвилини. Ця затримка потрібна для реально рандомної генерації.*

Код програм починається з наступної сторінки. Початок кожної програми виділено жирним червоним шрифтом.

Висновок: всі завдання було виконано й усі програми виконують свій функціонал — що від них потребується, те вони й виконують. Ми ознайомились з операторами циклів, принципами формування одновимірних, двовимірних, динамічних масивів, вивчили способи виділення пам'яті для динамічних масивів, навчились будувати математичну модель задачі, вивчили методи перевірки правильності вхідних даних, вивчили можливості оголошення та ініціалізації рядів, вивчили головні функції роботи з рядами, засвоїли використання головних функцій бібліотек та головне: отримали практичні навички в написанні програм.

```

#include <iostream>
#include <windows.h>
using namespace std;

// REVIEW
// ===== АВТОРСТВО =====
// -
// Написал код:
// Начинающий программист - Калиновский Валентин
// -
// ===== ВСТУПЛЕНИЕ =====
// -
// Здесь я написал код к следующим лабораторным заданиям:
// 1) Циклы (лаб 1 : Сем2 | Лаб 14 : Курс1) - задания ONE, TWO, THREE
// 2) Одномерные массивы (лаб 2 : Сем2 | Лаб 15 : Курс1) - задание FOUR
// 3) Двумерные массивы (лаб 3 : Сем2 | Лаб 16 : Курс1) - задания FIVE, SIX
// 4) Динамические массивы (лаб 4 : Сем2 | Лаб 17 : Курс1) - задание SEVEN
// 5) Обработка строк (лаб 5 : Сем2 | Лаб 18 : Курс1) - задание EIGHT
// -
// ===== НАСТРОЙКА =====
// ===== ВЫБЕРИТЕ НОМЕР КОДА ЗАДАНИЯ =====
// -
// Для выбора введите в поле #define под этим комментарием один из
// следующих номеров: ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT
// -
// =====
// CHOISE:

#define EIGHT

// NOTE: здесь описана функция для вывода двумерных массивов, чтобы не копировать
// её в каждый класс, где она нужна.
template <typename T>
void printMass(T Mass, int width, int height)
{
    for (int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
            cout << Mass[i][j] << " ";
        cout << "\n";
    }
}

#ifdef ONE
class One{
public:
    One()
    {
        while (i <= 300)
        {
            if (i % 2 == 0)
            {
                if (i % 13 == 0)
                {
                    if (i + 13 > 300 || i + 13 * 2 > 300)
                        cout << i;
                    else
                        cout << i << " + ";
                    Sum += i;
                }
            }
            i++;
        }
        cout << " = " << Sum;
    }

private:
    int Sum = 0, i = 1;
};
#endif //ONE

#ifdef TWO
class Two{
public:

```

```

Two()
{
    do {
        for (int j = 2; j < i; j++)
        {
            if (i % j == 0)
            {
                sampleInt = false;
                j = i;
            }
            else
                sampleInt = true;
        }
        if (sampleInt)
            cout << i << " ";
        i++;
    } while (i <= 300);
}

private:
    int i = 1;
    bool sampleInt;
};
#endif //TWO

```

```

#ifdef THREE
class Three{
public:
    Three()
    {
        for (int i = 1; i < 10; i++)
        {
            for (int j = i; j < 10; j++)
            {
                a = j;
                printResult(a, b, c);
            }
            a = i;
            for (int j = i; j < 10; j++)
            {
                b = j;
                printResult(a, b, c);
            }
            b = i;
            for (int j = i; j < 10; j++)
            {
                c = j;
                printResult(a, b, c);
            }
            c = i;
            for (int j = i; j < 10; j++)
            {
                a = j;
                b = j;
                printResult(a, b, c);
            }
            a = i;
            b = i;
            for (int j = i; j < 10; j++)
            {
                a = j;
                c = j;
                printResult(a, b, c);
            }
            a = i;
            c = i;
            for (int j = i; j < 10; j++)
            {
                b = j;
                c = j;
                printResult(a, b, c);
            }
            b = i;
            c = i;
            for (int j = i; j < 10; j++)

```

```

        {
            a = j;
            b = j;
            c = j;
            printResult(a, b, c);
        }
        a = i;
        b = i;
        c = i;
    }
}

private:
    int a = 1, b = 1, c = 1;
    static void printResult(int a, int b, int c)
    {
        if ((a * b) - (c * c * c) - 9 == 0)
            cout << "Результат: a = " << a << " , b = " << b << " , c = " << c << "\n";
    }
};
#endif //THREE

```

#ifdef FOUR

// NOTE: дополнительно мне нужны библиотеки для работы с генератором и временем -
#include <random>
#include <ctime>

```

class Four{
public:
    Four()
    {
        cout << "Как заполнить массив? Вы сами его введёте (1), или массив должен
сгенерироваться программно (2)?\n";
        cin >> answer;
        switch (answer)
        {
            case 1:
                for (int i = 0; i < 15; i++)
                {
                    cout << "\n" << i + 1 << " элемент массива: ";
                    cin >> Mass[i];
                }
                break;
            case 2:
                for (int i = 0; i < 15; i++)
                {
                    srand(time(NULL));
                    Sleep(1500);
                    Mass[i] = 1 + rand() % 18;
                    cout << "\n" << i + 1 << " элемент массива: " << Mass[i];
                }
                break;
            default:
                cout << "Введено неправильное значение! Ответ принимается за \"2\":\n";
                for (int i = 0; i < 15; i++)
                {
                    srand(time(NULL));
                    Sleep(1500);
                    Mass[i] = 1 + rand() % 18;
                    cout << "\n" << i + 1 << " элемент массива: " << Mass[i];
                }
                break;
        }
    }
};
// BUG: Почему то здесь при проверке значений цикл не хочет работать, если два
парных // элемента идут по соседству, например, если 13 и 14 элемент будут равны, он
их // всё равно в сумму не засчитает - он посчитает сумму для элемента 13 с
любыми // элементами, только если это не 12 и 14 элементы. Аналогично и для
других...
for (Num = 1; Num < 15; Num++)
{

```

```

        if (prov == Mass[Num])
        {
            Sum++;
        }
        if (Num == 14)
        {
            prov = Mass[counter];
            Num = counter + 1;
            counter++;
        }
    }
    cout << "\nСумма пар: " << Sum << "\n";
}

private:
    int Mass[15]{{}}, answer{{}}, Num = 0, prov = Mass[Num], Sum = 0, counter = 1;
};
#endif //FOUR

```

#ifdef FIVE

```

// NOTE: дополнительно мне нужны библиотеки для работы с генератором и временем -
#include <random>
#include <ctime>

class Five{
public:
    Five(int Mass[5][5])
    {
        cout << "Сначала нужно сгенерировать (5*5=) 25 элементов:\n";
        for (int & element : elements)
        {
            // NOTE: для красоты вида результата я решил ограничить диапазон с 1 - 25
            // 10 - 25. Волноваться не стоит: все эти элементы входя в множество
            // 1 - 25, а матрица выглядит ровной и красивой.
            srand(time(NULL));
            Sleep(1250);
            element = 10 + rand() % 15;
            cout << element << " ";
        }
        cout << "\nДальше, относительно схемы своего варианта, я заполняю двумерный
массив\n"
            "этими элементами в этой последовательности. Результат:\n";
        filling(elements, Mass, &num, 'y', '+', 5, 0, 0);
        filling(elements, Mass, &num, 'x', '+', 4, 4, 1);
        filling(elements, Mass, &num, 'y', '-', 4, 3, 4);
        filling(elements, Mass, &num, 'x', '-', 3, 0, 3);
        filling(elements, Mass, &num, 'y', '+', 3, 1, 1);
        filling(elements, Mass, &num, 'x', '+', 2, 3, 2);
        filling(elements, Mass, &num, 'y', '-', 2, 2, 3);
        filling(elements, Mass, &num, 1, 2);
        filling(elements, Mass, &num, 2, 2);
    }

private:
    int elements[25]{{}}, num = 0;
    static void filling(const int el[25], int mass[5][5], int *Num, char direction,
    char sign, int amount, int y, int x)
    {
        int z = 0;
        while (z < amount)
        {
            if (direction == 'x' && sign == '+')
            {
                mass[y][x] = el[*Num];
                x++;
                *Num = *Num + 1;
            }
            else if (direction == 'y' && sign == '+')
            {
                mass[y][x] = el[*Num];
                y++;
                *Num = *Num + 1;
            }
        }
    }
}

```



```

        else if (direction == 'x' && sign == '-')
        {
            mass[y][x] = el[*Num];
            x--;
            *Num = *Num + 1;
        }
        else if (direction == 'y' && sign == '-')
        {
            mass[y][x] = el[*Num];
            y--;
            *Num = *Num + 1;
        }
        z++;
    }
}
static void filling(const int el[25], int mass[5][5], int *Num, int y, int x)
{
    mass[y][x] = el[*Num];
    *Num = *Num + 1;
}
};
#endif //FIVE

```

```

#ifdef SIX
class Six{
public:
    explicit Six(int Mass[5][5])
    {
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                if (count % 2 == 0)
                    Mass[i][j] = 1;
                else
                    Mass[i][j] = 0;
                count++;
            }
        }
    }

private:
    int count = 0;
};
#endif //SIX

```

```

#ifdef SEVEN
// NOTE: дополнительно мне нужны библиотеки для работы с генератором и временем -
#include <random>
#include <ctime>

class Seven{
public:
    Seven()
    {
        cout << "\nВведите размеры массива:\nколичество строчек - ";
        cin >> height;
        cout << "Количество элементов каждой строки - ";
        cin >> width;
        length = height * width;
        if (height <= HEIGHT && width <= WIDTH)
        {
            cout << "\nС какого числа должен генерироваться числа для массива? ";
            cin >> intA;
            cout << "До какого? ";
            cin >> intB;
            intB -= intA;
            isStart = true;
        }
        else
        {
            cout << "Ошибка ввода: размер массива больше максимального!";
            isStart = false;
        }
    }
};
#endif //SEVEN

```

```

}
~Seven()
{
    for (int count = 0; count < height; count++)
        delete []Mass[count];
    delete []Mass;
    delete []Series;
}
void iswork()
{
    if (isStart)
        works();
}

private:
// NOTE: так как максимальные размеры не заданы - я сам их придумал и установил.
const int HEIGHT = 15, WIDTH = 15;
bool isStart;
int height{}, width{}, intA{}, intB{}, length{}, numSeries = 0, **Mass{},
*Series{};
void works()
{
    cout << "\nГенерируем начальную матрицу:\n";
    Mass = new int*[height];
    for (int count = 0; count < height; count++)
    {
        Mass[count] = new int[width];
        for (int j = 0; j < width; j++)
        {
            srand(time(NULL));
            sleep(1250);
            Mass[count][j] = intA + rand() % intB;
            cout << Mass[count][j] << " ";
        }
        cout << "\n";
    }
    // NB: во-первых, я предположу, что этот одномерный массив есть ряд, элементы
    // которого соответствуют так - {[0][0] = [0], [1][1] = [1], ...,
    // [0][4] = [4], [1][0] = [5], [1][1] = [6], ..., [4][4] = [24]}, и, во-
    // вторых, так как нигде в задании не был задан этот ряд, я предположу, что мне для его
    // выполнения нужно заново провести ту же работу, что и для двумерного массива, за
    // исключением, что это будет одномерный ряд и потом, как требуется, просто заменить старые
    // соответствующие элементы на новые соответствующие из нового ряда, а выводить нужно
    // первоначальный массив, новый ряд и новый массив. Так я понял и так поступлю.
    cout << "\nГенерируем новый одномерный массив с количеством элементов\n"
    "начальной матрицы: (" << height << "*" << width << "=") " << length <<
    " элементов.\n\n";
    "Все элементы генерируются в том же диапазоне.\n\nНовый ряд:\n";
    Series = new int[length];
    for (int i = 0; i < length; i++)
    {
        srand(time(NULL));
        sleep(1250);
        Series[i] = intA + rand() % intB;
        cout << Series[i] << " ";
    }
    cout << "\n\nЗаменяем каждый элемент начального массива соответствующими
    элементами из ряда.\n";
    for (int i = 0; i < height; i++)
        for (int j = 0; j < width; j++)
        {
            Mass[i][j] = Series[numSeries];
            numSeries++;
        }
    cout << "\nВыводим результат - новая матрица:\n";
    printMass(Mass, width, height);
}
};
#endif //SEVEN

```

```

#ifndef EIGHT
#include <string>
class Eight{
public:
    Eight()
    {
        cout << "Введите строку: ";
        cin.getline(userSTR, 140);
        countWord();
        wordSuggestion();
        checkDubbling();
    }
private:
    string word;
    char userSTR[140] {}, userSTRnew[140] {}, *userSTRword {};
    int countSymbol = 0, countWORD = 1;
    bool check {};
    void countWord()
    {
        for (int i = 0; i < sizeof(userSTR) / sizeof(char); i++)
            if (userSTR[i] == ' ' && userSTR[i - 1] != '-')
                countWORD++;
        cout << "\nКоличество слов в вашем тексте: " << countWORD;
    }
    void wordSuggestion()
    {
        for (char i : userSTR)
            if (i != '\0')
                countSymbol++;
        cout << "\nКоличество написанных символов: " << countSymbol;
    }
    void checkDubbling()
    {
        userSTRword = strtok(userSTR, " ");
        while (userSTRword != nullptr)
        {
            word = userSTRword;
            for (int j = 0; j < word.size(); j++)
            {
                if (userSTRword[j] == userSTRword[j + 1] && userSTRword[j + 1] != '.')
                {
                    check = false;
                    break;
                }
                else
                    check = true;
            }
            if (check)
                strcat(userSTRnew, userSTRword);
            strcat(userSTRnew, " ");
            userSTRword = strtok(nullptr, " ");
        }
        strcat(userSTRnew, "\0");
        cout << "\n\nНовое предложение без слов с удвоенными буквами:\n" <<
userSTRnew << "\n";
    }
};
#endif //EIGHT

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    char end[] = "\n\nКонец задания.\n";
#ifdef ONE
    cout << "Подготовил Калиновский Валентин.\nПервое задание (лабораторная №1):\n\n";
    One one;
    cout << end;
#endif //ONE
#ifdef TWO
    cout << "Подготовил Калиновский Валентин.\nВторое задание (лабораторная №1):\n\n";
    Two two;
    cout << end;
#endif //TWO
#ifdef THREE
    cout << "Подготовил Калиновский Валентин.\nТретье задание (лабораторная №1):\n\n";

```

```

    Three three;
    cout << end;
#endif //THREE
#ifdef FOUR
    cout << "Подготовил Калиновский Валентин.\nЗадание по лабораторной №2:\n\n";
    Four four;
    cout << end;
#endif //FOUR
#ifdef FIVE
    cout << "Подготовил Калиновский Валентин.\nПервое задание (лабораторная №3):\n\n";
    int Mass[5][5];
    Five five(Mass);
    printMass(Mass, 5, 5);
    cout << end;
#endif //FIVE
#ifdef SIX
    cout << "Подготовил Калиновский Валентин.\nВторое задание (лабораторная №3):\n\n";
    int Mass[5][5];
    Six six(Mass);
    printMass(Mass, 5, 5);
    cout << end;
#endif //SIX
#ifdef SEVEN
    cout << "Подготовил Калиновский Валентин.\nЗадание по лабораторной №4:\n\n";
    Seven seven;
    seven.isWork();
    cout << end;
#endif //SEVEN
#ifdef EIGHT
    cout << "Подготовил Калиновский Валентин.\nЗадание по лабораторной №5:\n\n";
    Eight eight;
    cout << end;
#endif //EIGHT
    cin.get();
    cin.get();
    return 0;
}

```