|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**О Т Ч Е Т**

по лабораторной работе №\_3\_

Вариант № 25

**Дисциплина**: Технология разработки программных систем \_

**Название**: Оценка эффективности и качества программ \_



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-42б |  | 16.05.2021 | И.С. Марчук |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е.К. Пугачев |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |

*2021 г.*

**Цель работы –** изучить основные критерии оценки и способы повышения эффективности и качества программных продуктов.

**Задание:** Вариант 25

написать программу, которая создает случайным образом двумерный массив целых чисел, а затем обнуляет элементы ниже главной диагонали.

**1. Исходная программа**

В листинге 1 предоставлена исходная программа.

Листинг 1 – исходный код

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <time.h>

int main()

{

int N=10,j, i, sum = 0;

srand(time(NULL));

int\*\* a = (int\*\*)malloc(sizeof(int\*) \* N);

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

a[i] = (int\*)malloc(sizeof(int\*) \* N);

for (i = 0; i<N; i++){

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

a[i][j] = -30 + rand()%(30 + 30 + 1);

}

}

printf("\nИсходный массив:\n");

for (i = 0; i<N; i++){

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

printf("%3d\t", a[i][j]);

}

printf("\n");

}

int\*\* b = (int\*\*)malloc(sizeof(int\*) \* N);

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

b[i] = (int\*)malloc(sizeof(int\*) \* N);

for (i = 0; i<N; i++){

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

if (i == j){

b[i][j] = a[i][j];

}

}

}

for (i = 0; i <N; i++){

if (i == 0){

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

b[i][j] = a[i][j];

}

if (i == 1){

for (j = 1; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<1; j++)

b[i][j] = 0;

}

if (i == 2){

for (j = 2; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<2; j++)

b[i][j] = 0;

}

if (i == 3){

for (j = 3; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<3; j++)

b[i][j] = 0;

}

if (i == 4){

for (j = 4; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<4; j++)

b[i][j] = 0;

}

if (i == 5){

for (j = 5; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<5; j++)

b[i][j] = 0;

}

if (i == 6){

for (j = 6; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<6; j++)

b[i][j] = 0;

}

if (i == 7){

for (j = 7; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<7; j++)

b[i][j] = 0;

}

if (i == 8){

for (j = 8; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<8; j++)

b[i][j] = 0;

}

if (i == 9){

for (j = 9; j<N; j++)

b[i][j] = a[i][j];

for (j = 0; j<9; j++)

b[i][j] = 0;

}

}

printf("\nИзмененный массив:\n");

for (i = 0; i<N; i++){

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

printf("%3d\t", b[i][j]);

}

printf("\n");

}

return 0;

}

В ходе анализа, были выявлены и проанализированы недочеты исходной программы план их устранения предоставлен в таблице 1.

Таблица 1 – план улучшения программы

|  |  |
| --- | --- |
| № | Содержание |
| 1 | циклы инициализации массива и вывода на печать можно объединить |
| 2 | тип элементов массива int можно заменить на short, и в итоге получить бОльшую экономию памяти |
| 3 | количество элементов (переменная N) не может быть отрицательным числом, а следовательно, можно тип переменной заменить, например, на unsigned int и сэкономить память |
| 4 | можно удалить неиспользуемую переменную |
| 5 | в конце выполнения программы не очищается выделенная память – можно добавить функцию free() |
| 6 | можно повысить универсальность, если размерность матрицы и диапазон генерируемых данных, и количество данных задавать с клавиатуры |
| 7 | проверяемость результатов работы программы имеет высокий уровень, но только при небольших размерах массива |
| 8 | хороший стиль программирования предполагает использование комментариев, а в исходной программе их явно не хватает |

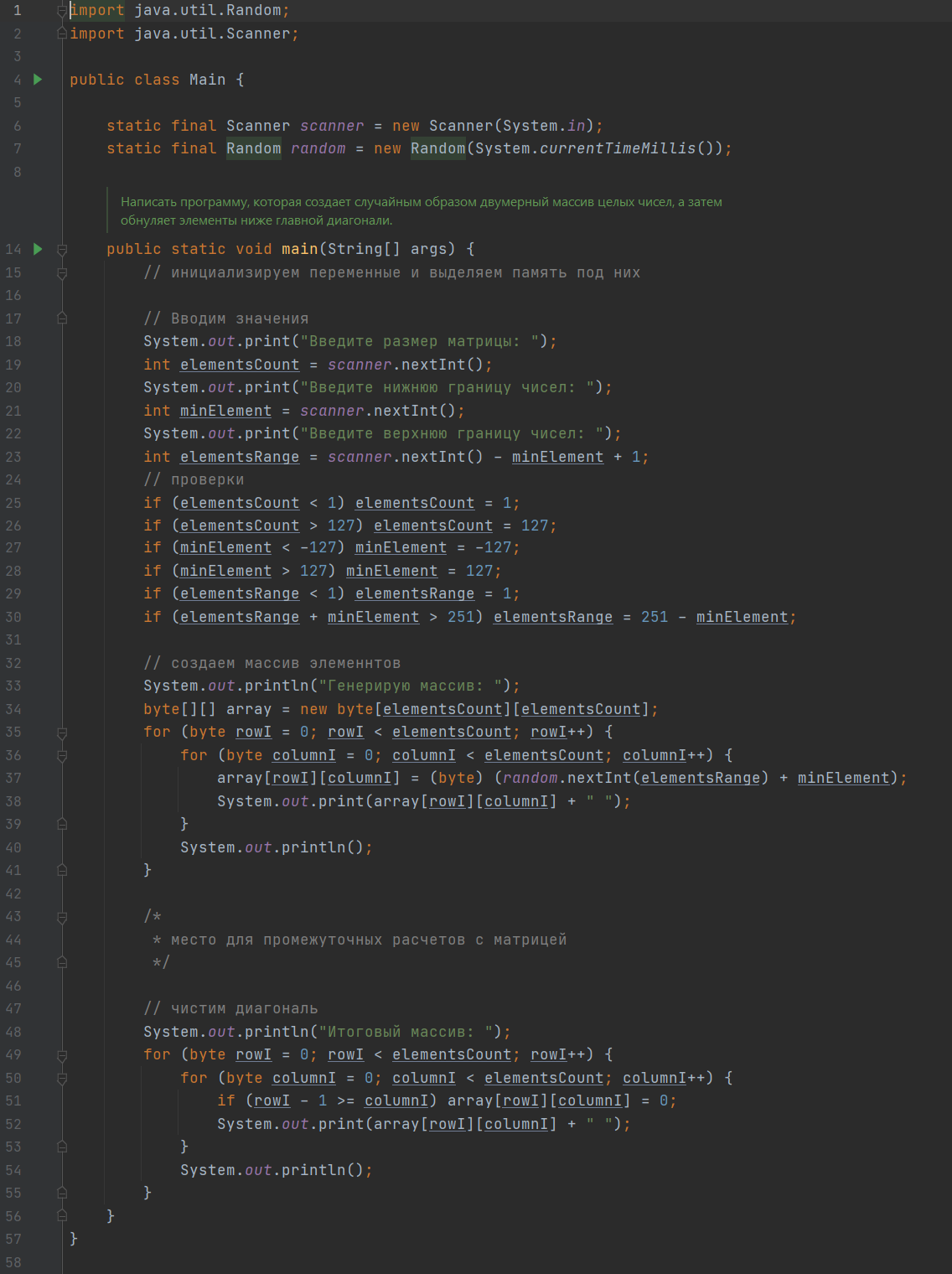
В итоге можно сделать много изменений в исходной программе и получить более эффективную и качественную программу.

Ниже представлена программа, которая является модифицированной версией исходной программы.

**2. Улучшенная программа**

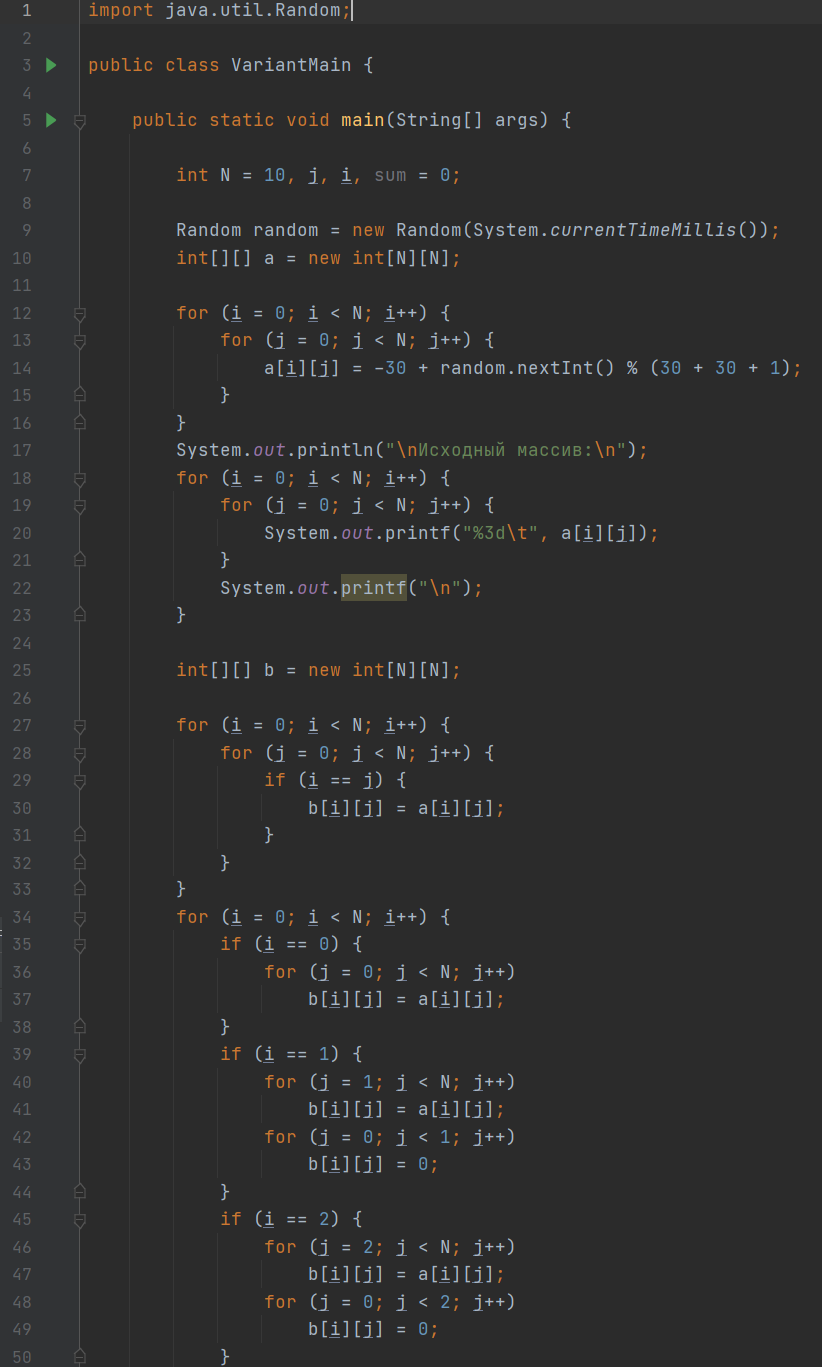
В листинге 2 предоставлен код доработанной программы на языке Java.

Листинг 2 – улучшенная программа

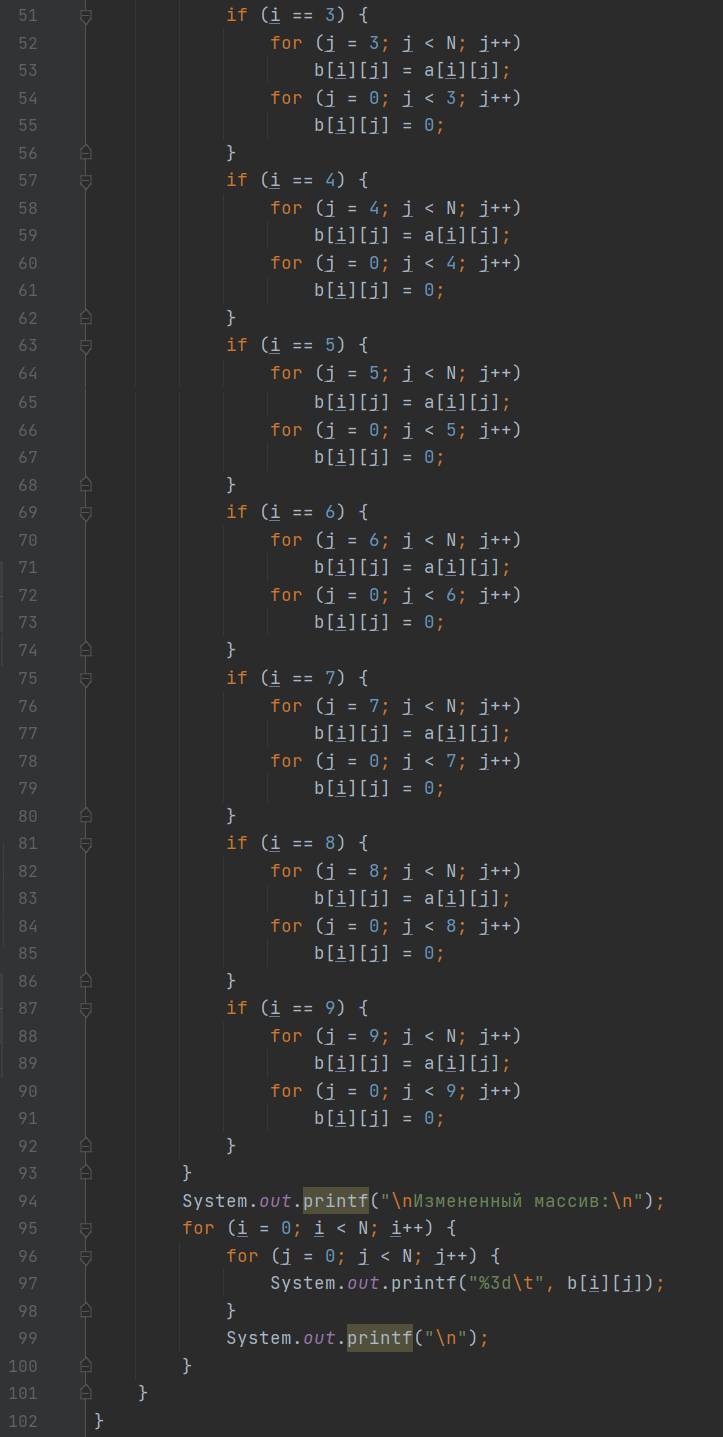
****

Для того чтобы написанную программу можно было сравнивать с исходной программой, текст исходной программы был переписан на язык Java с сохранением структуры переменных и кода. Получившаяся программа представлена в листинге 3.

Листинг 3 – Текст исходной программы на языке Java

****

Продолжение листинга 3:

****

**Оценка эффективности**

В таблице 2 отражены результаты замеров времени и оценки памяти для исходной программы и улучшенной, а также указаны недостатки и способы улучшения.

Таблица 2 – Оценка эффективности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий оценки | Исходная программа | | Улучшенная программа | |
| Недостатки | Количественная оценка | Улучшения | Количественная оценка |
| Время выполнения | 1) Цикл инициализации матрицы и ее вывода на экран можно объединить.  2) Цикл переприсваивания можно убрать, т.к. вторая матрица не нужна.  3) Вместо одного цикла выполняется 10 по условию.  Многократное обращение к элементам матрицы. | 31.262 мс  27.170 мс  32.694 мс | 1) объединены циклы инициализации и вывода на экран исходной матрицы.  2) удалена вторая матрица и весь связанный с ней код.  3) 10 циклов объединены в один и заменены дополнительным условием в цикле вывода обработанной матрицы. | 7.823 мс  7.968 мс  7.885 мс |
| Оперативная память | Слишком большой тип переменных и элементов массива.  В исходной программе не использовалась очистка памяти, в язык java эта функция встроена автоматически.  Лишний двумерный массив. Объявлена неиспользуемая переменная. | 4 переменные типа int: .  Две матрицы по 2500 элементов типа int:  Итого 20016 байт. | Изменен тип элементов. Память, как уже говорилось чистится автоматически. Удалены лишний массив и переменная. | 3 переменных типа int.  2 переменных типа byte .  2500 элементов матрицы типа byte:  Итого 2514 байт. |

Изначально размерность матрицы задана 10, но замеры времени и памяти производились при размерности матрицы, равной 50, для наглядности.

В итоге после улучшения программы, мы получили экономию времени выполнения в 3.82 раза и экономию оперативной памяти в 7,96 раз.

**Оценка качества**

В таблице 3 отражены результаты оценки качества исходной программы.

Таблица 3 – Оценка качества

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Результаты оценки | Критерии оценки | | | |
| Правильность | Универсальность | Проверяемость | Точность результатов |
| Недостатки | Программа работает согласно условию задачи. | Все данные заданы в коде программы или генерируются. Пользователь не может вводить нужные ему данные не имея доступа к сходному коду программы. | При большом размере матрицы проверка затруднительна, т.к. матрица не умещается в терминале. | Поскольку по заданию элементы должны были вводиться случайно, а очиска нижней части матрицы происходит корректно. То выданный программой результат вполне можно считать как точный. |
| Оценка, баллы | 5/5 | 3/5 | 3/5 | 5/5 |

**Заключение**

В результате проведенных экспериментов были выполнены замеры времени работы программы, оценки памяти, а также предложены способы повышения эффективности программы и улучшения качества написанного кода.