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		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: 25. Написать программу, которая создает случайным образом массив записей. Каждая запись имеет поля: год, месяц, день и описание события. Программа должна сортировать по полю «дата» (программа v25.dpr).

1. Исходная программа

В листинге 1 предоставлена исходная программа.

Листинг 1 – исходный код

```
program V25;
{$APPTYPE CONSOLE}
uses SysUtils;

Const N=7;
Type
Tzap=record
  y, m, d: integer;
  e: string[20];
end;
Ts = array [1..N] of Tzap;

Function Diap(n,k:integer):integer;
begin
  Diap:=random(k-n)+n;
end;

Function st(n: byte): string;
  var i, k: integer; s: string;
begin
  s:='';
  k:=random(n)+1;
  for i:=1 to k do
    s:=s+chr(random(25)+65);
  st:=s;
end;

Var i,j:integer; z:Tzap; M:Ts;
begin
  randomize;
  for i:=1 to N do
  begin
    M[i].y:=diap(2000,2019);
    M[i].m:=diap(1,12);
    M[i].d:=diap(1,31);
    M[i].e:=st(20);
  end;

  for i:=1 to N do
    writeln(i,'. ',M[i].y:6,M[i].m:5,M[i].d:5,'    ',M[i].e);
  writeln;

  for i:=1 to N do
    for j:=1 to N-1 do
      if M[j].y< M[j+1].y then
      begin
        z:=M[j];M[j]:=M[j+1]; M[j+1]:=z;
      end;

  for i:=1 to N do
    writeln(i,'. ',M[i].y:6,M[i].m:5,M[i].d:5,'    ',M[i].e);

  readln;
end.
```

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

Таблица 1 – план улучшения программы

№	Содержание
1	Программа сортирует записи только по году, а не по дате.
2	При генерации не учитывается, что в месяце дней может быть меньше чем 31.
3	Можно повысить универсальность, если диапазоны генерируемых данных и количество записей будут вводиться с клавиатуры.
4	Для переменных месяца и дня в структуре можно использовать типы меньше, например byte
5	в конце выполнения программы не очищается выделенная память – можно добавить вызов процедуры FreeMem()
6	Сортировку пузырьком можно заменить на более быструю
7	Переменным можно дать название, отражающее их назначение.
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

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Введите год начала:");
    int startYear = scanner.nextInt();
    if (startYear < 0) {System.out.println("Год не может быть отрицательным");return;}
    System.out.print("Введите год конца:");
    int endYear = scanner.nextInt();
    if (endYear < 0) {System.out.println("Год не может быть отрицательным");return;}
    if (startYear > endYear) {System.out.println("Год начала не может быть больше года конца");return;}
    System.out.print("Введите количество записей:");
    int eventsCount = scanner.nextInt();

    Tzap[] eventList = new Tzap[eventsCount];
    Random random = new Random( seed: 10);
    int i;
    for (i = 0; i < eventsCount; i++) {
        eventList[i] = generateEvent(startYear, endYear, random);
    }

    for (i = 0; i < eventsCount; i++)
        System.out.printf("%d. %6d%5d%5d    %s\n", i + 1, eventList[i].y, eventList[i].m, eventList[i].d, eventList[i].e);
    System.out.println();

    // сортировка слиянием
    mergeSort(eventList,  left: 0,  right: eventList.length - 1);

    for (i = 0; i < eventsCount; i++)
        System.out.printf("%d. %6d%5d%5d    %s\n", i + 1, eventList[i].y, eventList[i].m, eventList[i].d, eventList[i].e);
    System.out.println();
}
```

Листинг 3 – Новый метод сортировки (сортировка слиянием)

```
100     static void mergeSort(Tzap[] source, int left, int right) {
101         // Выберем разделитель, т.е. разделим пополам входной массив
102         int delimiter = left + ((right - left) / 2) + 1;
103 
104         // Выполним рекурсивно данную функцию для двух половинок (если сможем разбить)
105         if (delimiter > 0 && right > (left + 1)) {
106             mergeSort(source, left,  right: delimiter - 1);
107             mergeSort(source, delimiter, right);
108         }
109         // Создаём временный массив с нужным размером
110         Tzap[] buffer = new Tzap[right - left + 1];
111         // Начиная от указанной левой границы идём по каждому элементу
112         int cursor = left;
113         int mid = delimiter;
114         for (int i = 0; i < buffer.length; i++) {
115             // Мы используем delimeter чтобы указывать на элемент из правой части
116             // Если delimeter > right, значит в правой части не осталось недобавленных элементов
117             if (cursor < mid && (delimiter > right || compareDate(source[cursor], source[delimiter]) > 0)) {
118                 buffer[i] = source[cursor];
119                 cursor++;
120             } else {
121                 buffer[i] = source[delimiter];
122                 delimiter++;
123             }
124         }
125         System.arraycopy(buffer,  srcPos: 0, source, left, buffer.length);
126     }
127 
128 @   static int compareDate(Tzap a, Tzap b) {
129         // сравниваем год
130         if (a.y > b.y) return 1;
131         if (a.y < b.y) return -1;
132 
133         // если год одинаковый, сравниваем месяц
134         if (a.m > b.m) return 1;
135         if (a.m < b.m) return -1;
136 
137         // если год и месяц одинаковые, сравниваем дни (в случае одинаковых дат вернет 0)
138         return Byte.compare(a.d, b.d);
139     }
140 }
```

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 4.

Листинг 4 – Текст исходной программы на языке Java

```java
package com.company;
import java.util.Random;
public class Main {

    static Random random;

    static final int N = 7;

    static class Tzap {
        int y, m, d;
        String e;
    }

    static int diap(int n, int k) {
        return random.nextInt( bound: k - n) + n;
    }

    static String st(byte n) {
        int i, k;
        String s;

        s = "";
        k = random.nextInt(n) + 1;
        for (i = 1; i <= k; i++)
            s = s + (char) (random.nextInt( bound: 25) + 65);
        return s;
    }

    public static void main(String[] args) {
        int i, j;
        Tzap z;
        Tzap[] M = new Tzap[N];

        random = new Random();

        for (i = 0; i < N; i++) {
            M[i] = new Tzap();
            M[i].y = diap( n: 2000,  k: 2019);
            M[i].m = diap( n: 1,  k: 12);
            M[i].d = diap( n: 1,  k: 31);
            M[i].e = st((byte) 20);
        }

        for (i = 0; i < N; i++)
            System.out.printf("%d. %6d%5d%5d    %s\n", i + 1, M[i].y, M[i].m, M[i].d, M[i].e);
        System.out.println();

        // сортировка пузырьком
        for (i = 0; i < N; i++)
            for (j = 0; j < N - 1; j++)
                if (M[j].y < M[j + 1].y) {
                    z = M[j];
                    M[j] = M[j + 1];
                    M[j + 1] = z;
                }

        for (i = 0; i < N; i++)
            System.out.printf("%d. %6d%5d%5d    %s\n", i + 1, M[i].y, M[i].m, M[i].d, M[i].e);
        System.out.println();
    }
}
```

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	6)Сортировку пузырьком можно заменить на более быструю	100,700 мкс 88,000 мкс 91,600 мкс	1) Реализована сортировка MergeSort	63,000 мкс 63,400 мкс 61,800 мкс
Оперативная память	4)Для переменных месяца и дня в структуре можно использовать типы меньше, например byte 5)В конце выполнения программы не очищается выделенная память – можно добавить вызов процедуры FreeMem()	7 записей: - 3 переменные типа word - строка 20 символов; (2 + 3 * 2+1+20 * 2) *7 = 343 байта + 4 переменные по 4 байта = 359 байтов	Изменен тип элементов. Память чистится автоматически встроенным в Java механизмом.	7 записей: - 1 переменная типа word - 2 переменных типа byte - строка 20 символов; (2+2+2*1+1+20 * 2) * 7 = 329 байтов + 2 переменные по 4 байта = 337 байтов

В итоге после улучшения программы, мы получили экономию времени выполнения в 1,5 раза и экономию оперативной памяти в 1,2 раза.

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	1)Программа сортирует записи только по году, а не по дате. 2)При генерации не учитывается, что в месяце дней может быть меньше чем 31. Программа выдает неверные данные при сортировке	Все данные заданы в коде программы или генерируются. Пользователь не может вводить нужные ему данные не имея доступа к сходному коду программы. Количество записей фиксировано	Результаты отлично просматрива-ются в консоли, содержимое записей выводится корректно	В данном случае не оценивалась
Оценка, баллы	2/5	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.