Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Практична робота №2

Економіка ІТ-індустрії

Тема: Метрики розміру. Mipa Line Of Code

Виконав

студент групи ІП-11:

Панченко С. В.

Перевірив:

Родіонов П. В.

Київ 2024

Зміст

1 МЕТА ПРАКТИЧНОЇ РОБОТИ

Ознайомитися з загальними поняттями щодо вимірювань та метрикою розміру з мірою Lines of Code. Напрацювати вміння застосування засобів вимірювання метрики. Отримати загальні вміння щодо застосування метрики в економіці програмного забезпечення.

2 ЗАВДАННЯ

1. Застосовуючи вимірювачі у відповідних середовищах програмування (Visual Studio, Code Counter for Java, CodeCounter, та інші), на прикладі власних програмних текстів виконати вимірювання розміру.
2. Здійснити відповідні економічні розрахунки.
3. Дослідити рівні мов програмування C# та Java.
4. Захистити виконану роботу.

3 ВИКОНАННЯ

3.1 Метрика власного проекту

Розглянемо написаний мною проект FlaskProject [1]. Цей застосунок надає можливість керувати різними відділеннями людей, що спеціалізуються на конкретній мові програмування. Детальніше про проект можна дізнатися з документації.

У IDE PyCharm [2] завантажимо плагін Statistic [3], який надасть статистику LOC для всього проекту. Розглянемо на рисунках 3.1, 3.2, 3.3 статистику.

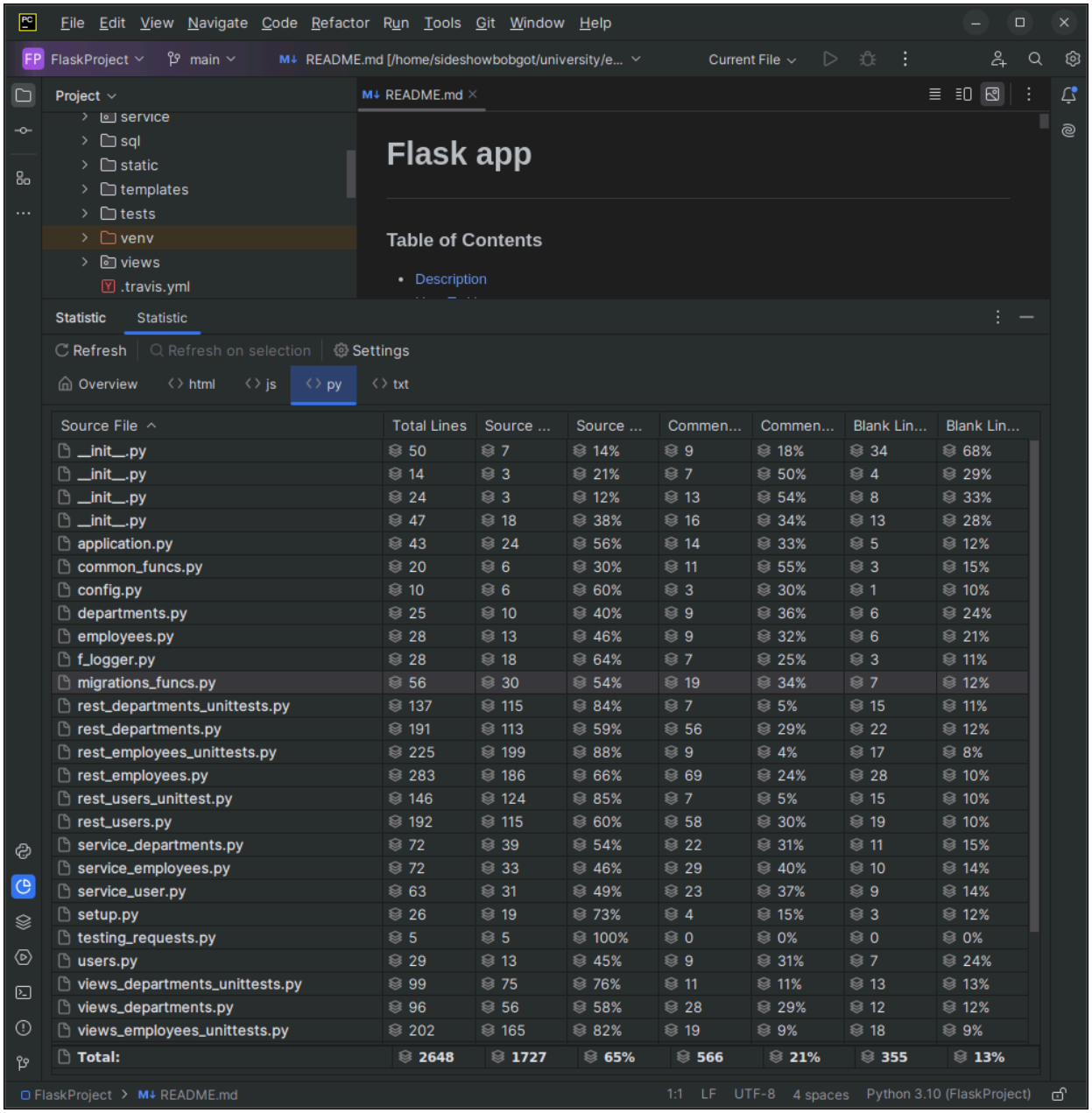

Рисунок 3.1 — Статистика для файлів з розширенням .py

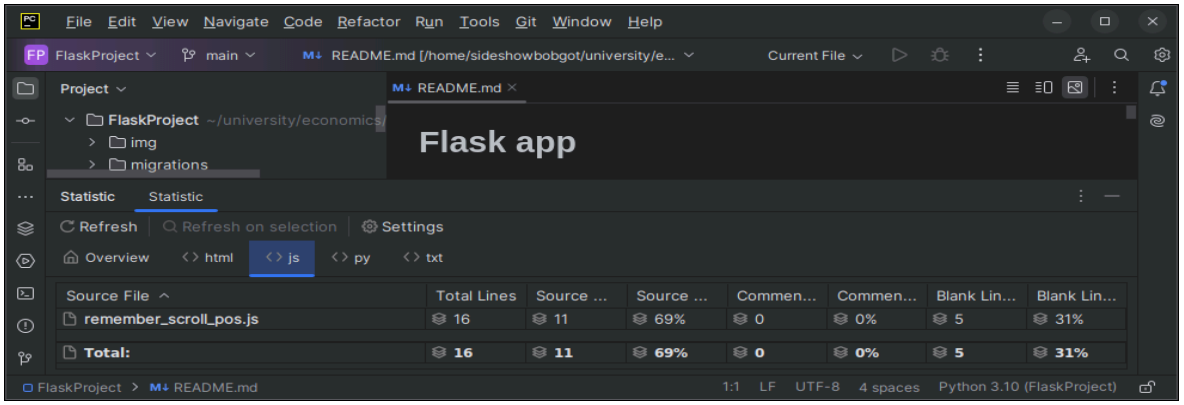

Рисунок 3.2 — Статистика для файлів з розширенням .js

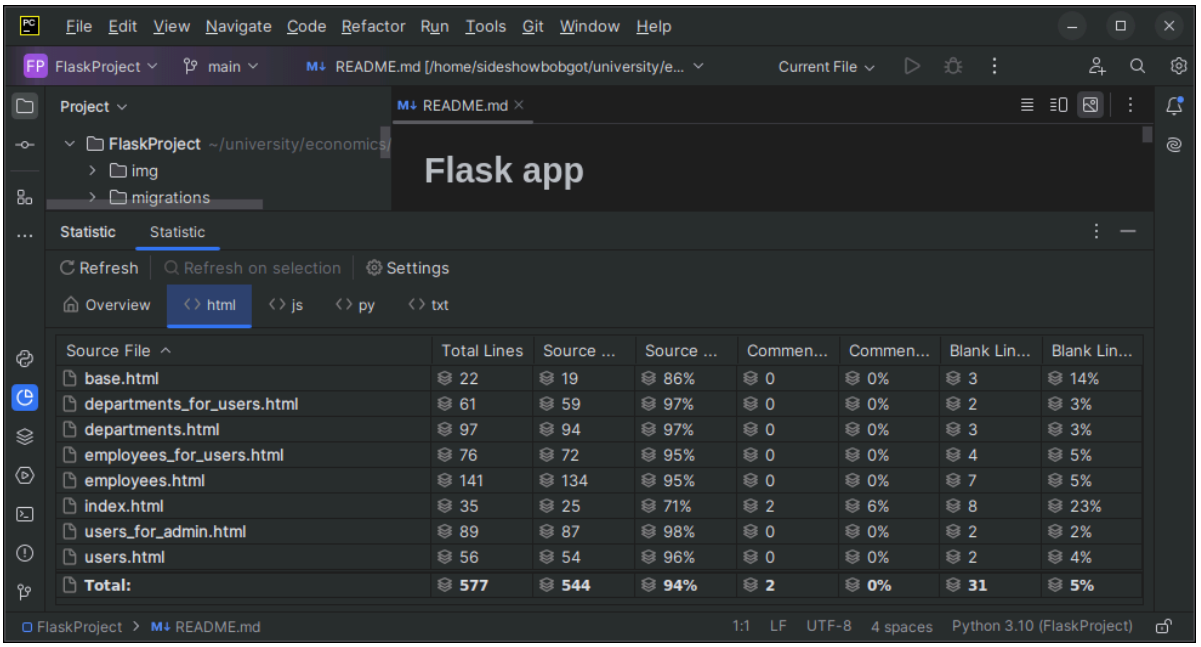

Рисунок 3.3 — Статистика для файлів з розширенням .html

На рисунку 3.4 розглянемо загальну статистику.

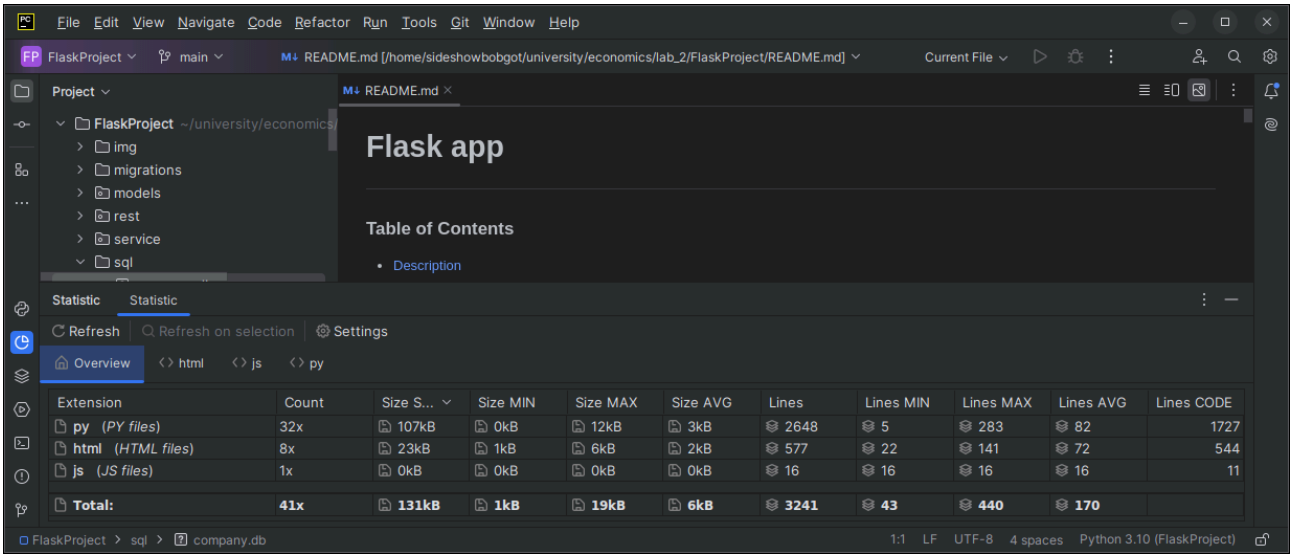

Рисунок 3.4 — Загальна статистика проєкту

У проекті є 3241 рядок коду. Більшість коду написано на Python (2648 рядків), що вказує на те, що основна логіка додатку знаходиться на серверній

стороні. HTML файли мають 577 рядків, що говорить про простий інтерфейс без складних елементів. JavaScript використовується дуже мало — лише 16 рядків.

З цього випливає, що проєкт відноситься до типу Organic, оскільки кількість рядків менша 25 тисяч.

3.2 Економічні розрахунки Organic-проекту

З таблиці 3.1 візьмемо коефіцієнти для Organic проекту.

Таблиця 3.1 — Коефіцієнти для проектів різного типу

Тип проекту	ab	bb	cb	db
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Розрахуємо зусилля:

$$\text{Зусилля} = ab * \text{розмір}^{bb} = 2.4 * 3.241^{1.05} = 8,249435368 \frac{\text{людина}}{\text{місяць}}$$

Розрахуємо вартість:

$$\text{Вартість} = \text{Зусилля} * \text{зарплата}$$

$$\text{Вартість} = 8,249435368 * 20000\text{грн} = 164988,70736 \frac{\text{людина} * \text{грн}}{\text{місяць}}$$

Розрахуємо час на розробку:

$$\text{Часнарозробку} = cb * \text{Зусилля}^{db} = 2.5 * 8,249435368^{0.38} = 5,57418289$$

Проаналізуємо розраховані метрики. У реальності проект розробляв протягом 1.5 місяці лише я, у свій вільний час за 0 грн. Суттєва різниця з теоретичними даними може полягати у тому, що проєкт був написаний на мові Python[4], яка є досить легкою вивченні, а також за допомогою неї можна швидко писати невеликі проекти. Код для HTML [5] та JavaScript [6] брався зі стоку, тобто у більшості був написаний до мене.

3.3 Дослідження рівнів мов програмування C# та Java

Для дослідження рівнів мов програмування C# та Java напишемо просте

сортування бульбашкою. Код наведено у додатку А. Для генерування коду асемблера звернемося до сайту https://godbolt.org/. На рисунку 3.5 та 3.6 можна побачити його використання для C# та Java відповідно.

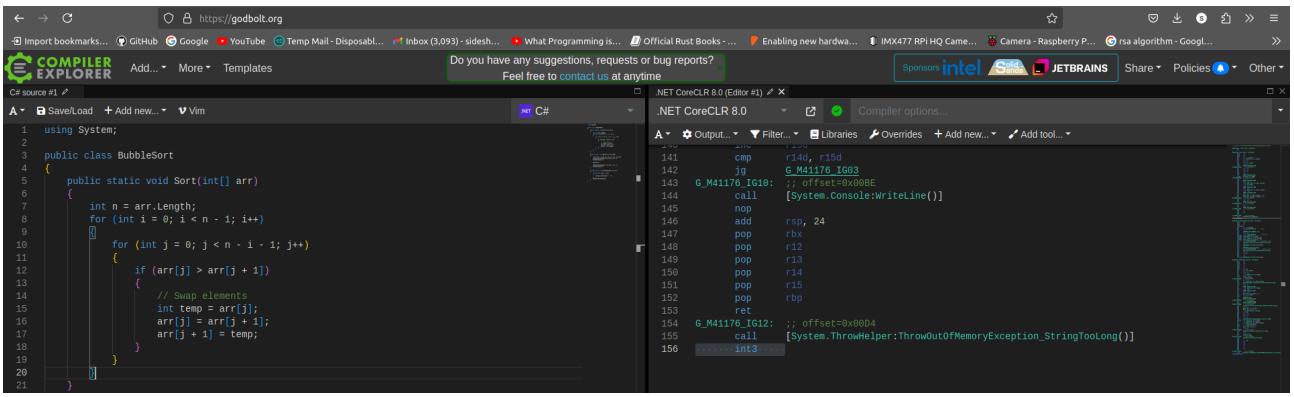

Рисунок 3.5 — Асемблер мови C#

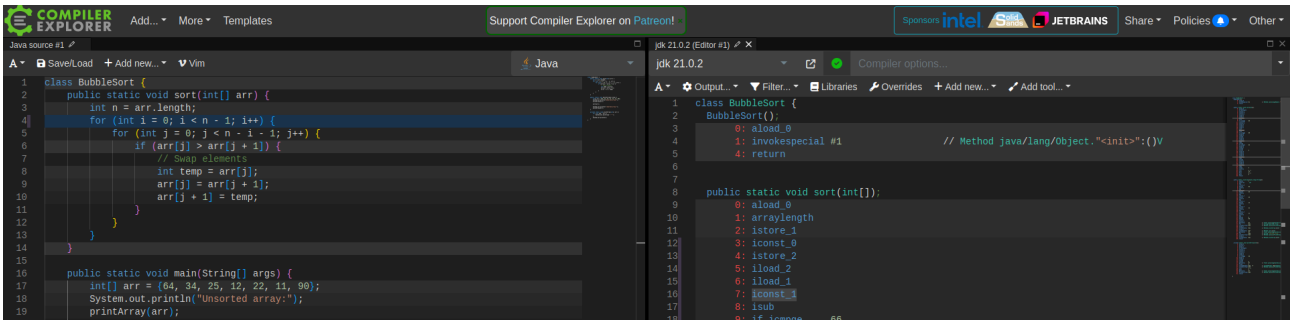

Рисунок 3.6 — Байт-код мови Java

Сайт надає можливість відразу поглянути, як ВМ машина C# транслюватиме байт-код під конкретну архітектуру, тобто в асемблер, але, на жаль, такої можливості для Java немає, тому розглянемо байт-код.

Розрахуємо співвідношення рядків асемблера(байт-коду) до звичайного коду.

$$C\# = \frac{156}{43} = 3,627906977$$

$$Java = \frac{134}{33} = 4,060606061$$

У результаті отримали доволі невеликі співвідношення кількості коду. У великих проектах з великим нашаруванням рівнів абстракції відношення може бути суттєво більше. У даному випадку декілька циклів та перестановок чисел у масиві дуже добре оптимізуються, бо це доволі типові операції.

ВИСНОВОК

У ході виконання практичної роботи було досягнуто поставлену мету - ознайомлення з метрикою розміру програмного забезпечення Lines of Code (LOC) та її застосуванням в економіці IT-індустрії. За допомогою плагіну Statistic в IDE PyCharm було проведено вимірювання розміру власного проекту FlaskProject, який містить 3241 рядок коду, більшість з яких (2648) написано мовою Python. На основі цих даних було визначено, що проєкт відноситься до типу Organic, оскільки кількість рядків коду менша за 25 тисяч.

Проведені економічні розрахунки для Organic-проекту показали, що теоретичні зусилля на розробку складають 8,25 людино-місяців, вартість - 164 988,71 грн, а час на розробку - 5,57 місяців. Однак, було виявлено значну різницю між теоретичними розрахунками та реальним досвідом розробки проекту, який тривав лише 1,5 місяці. Ця різниця пояснюється використанням мови Python, яка дозволяє швидко розробляти невеликі проекти, а також застосуванням готових рішень для HTML та JavaScript.

В рамках дослідження рівнів мов програмування було проаналізовано C# та Java. Для обох мов було реалізовано алгоритм сортування бульбашкою та проаналізовано згенерований асемблерний код (для C#) та байт-код (для Java). Розраховані співвідношення рядків асемблера/байт-коду до вихідного коду склали 3,63 для C# та 4,06 для Java. Ці співвідношення виявилися досить невеликими, що пояснюється простотою реалізованого алгоритму та ефективною оптимізацією типових операцій компіляторами. Важливо зазначити, що у великих проектах з більшим рівнем абстракції це співвідношення може бути суттєво більшим.

Практична робота дозволила отримати цінні навички застосування метрики LOC для оцінки розміру програмного забезпечення та проведення відповідних економічних розрахунків. Було виявлено обмеження даної метрики та фактори, які можуть впливати на точність прогнозів. Дослідження C# та Java в контексті метрики LOC підкреслює важливість врахування особливостей конкретних мов

програмування та рівня абстракції при оцінці розміру та складності проектів. Отримані результати демонструють, що навіть для схожих за синтаксисом мов високого рівня можуть бути відмінності у співвідношенні LOC до машинного коду. Ці знання та навички можуть бути корисними при плануванні та оцінці майбутніх проектів розробки програмного забезпечення, а також для розуміння економічних аспектів IT-індустрії з урахуванням специфіки різних мов програмування та рівнів абстракції.

ПОСИЛАННЯ

1. FlaskProject [Електронне джерело] - https://github.com/SideShowBoBGOT/FlaskProject
2. PyCharm [Електронне джерело] - https://www.jetbrains.com/pycharm/
3. Плагін Statistic [Електронне джерело] - https://plugins.jetbrains.com/plugin/4509-statistic
4. Python [Електронний ресурс] - https://www.python.org/
5. HTML [Електронний ресурс] - https://en.wikipedia.org/wiki/HTML
6. JavaScript [Електронний ресурс] - https://en.wikipedia.org/wiki/JavaScript

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студента групи ІП-11 4 курсу

Панченка С. В

Сортування бульбашкою мовою C#

```
using System;

public class BubbleSort
{
    public static void Sort(int[] arr)
    {
        int n = arr.Length;
        for (int i = 0; i < n - 1; i++)
        {
            for (int j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    // Swap elements
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void Main(string[] args)
    {
        int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
        Console.WriteLine("Unsorted array:");
        PrintArray(arr);

        Sort(arr);

        Console.WriteLine("\nSorted array:");
        PrintArray(arr);
    }

    private static void PrintArray(int[] arr)
    {
        foreach (int num in arr)
        {
            Console.Write(num + " ");
        }
        Console.WriteLine();
```

```
    }
}
```

Сортування бульбашкою мовою асемблер віртуальною машиною .NET CoreCLR 8.0

// coreclr 8.0.5+087e15321bb712ef6fe8b0ba6f8bd12facf92629

```
BubbleSort:.ctor():this (FullOpts):
       ret

BubbleSort:Sort(int[]) (FullOpts):
       push     rbp
       push     r15
       push     rbx
       lea      rbp, [rsp+0x10]
       mov      eax, dword ptr [rdi+0x08]
       mov      ecx, eax
       xor      edx, edx
       lea      esi, [rcx-0x01]
       test     esi, esi
       jle      SHORT G_M33098_IG08
G_M33098_IG03:  ;; offset=0x0017
       xor      r8d, r8d
       mov      r9d, ecx
       sub      r9d, edx
       dec      r9d
       test     r9d, r9d
       jle      SHORT G_M33098_IG07
       align    [0 bytes for IG04]
G_M33098_IG04:  ;; offset=0x0028
       cmp      r8d, eax
       jae      SHORT G_M33098_IG09
       mov      r10d, r8d
       mov      r10d, dword ptr [rdi+4*r10+0x10]
       lea      r11d, [r8+0x01]
       cmp      r11d, eax
       jae      SHORT G_M33098_IG09
       mov      ebx, r11d
       mov      r15d, dword ptr [rdi+4*rbx+0x10]
       cmp      r10d, r15d
       jle      SHORT G_M33098_IG06
       mov      r8d, r8d
```

```
       mov      dword ptr [rdi+4*r8+0x10], r15d
       mov      dword ptr [rdi+4*rbx+0x10], r10d
G_M33098_IG06:  ;; offset=0x0058
       mov      r8d, r11d
       cmp      r9d, r8d
       jg       SHORT G_M33098_IG04
G_M33098_IG07:  ;; offset=0x0060
       inc      edx
       cmp      edx, esi
       jl       SHORT G_M33098_IG03
G_M33098_IG08:  ;; offset=0x0066
       pop      rbx
       pop      r15
       pop      rbp
       ret
G_M33098_IG09:  ;; offset=0x006B
       call     CORINFO_HELP_RNGCHKFAIL
       int3

BubbleSort:Main(System.String[]) (FullOpts):
       push     rbp
       push     rbx
       push     rax
       vzeroupper
       lea      rbp, [rsp+0x10]
       mov      rdi, 0x76E747AF8328      ; int[]
       mov      esi, 7
       call     CORINFO_HELP_NEWARR_1_VC
       mov      rbx, rax
       mov      rdi, 0x76E7C6A30D38      ; const ptr
       vmovdqu  xmm0, xmmword ptr [rdi]
       vmovdqu  xmmword ptr [rbx+0x10], xmm0
       vmovdqu  xmm0, xmmword ptr [rdi+0x0C]
       vmovdqu  xmmword ptr [rbx+0x1C], xmm0
       mov      rdi, 0x76E7440F8618      ; 'Unsorted array:'
       call     [System.Console:WriteLine(System.String)]
       mov      rdi, rbx
       call     [BubbleSort:PrintArray(int[])]
       mov      rdi, rbx
       call     [BubbleSort:Sort(int[])]
       mov      rdi, 0x76E7440F8650      ; ' Sorted array:'
       call     [System.Console:WriteLine(System.String)]
```

```
       mov      rdi, rbx
       add      rsp, 8
       pop      rbx
       pop      rbp
       tail.jmp [BubbleSort:PrintArray(int[])]

BubbleSort:PrintArray(int[]) (FullOpts):
       push     rbp
       push     r15
       push     r14
       push     r13
       push     r12
       push     rbx
       sub      rsp, 24
       lea      rbp, [rsp+0x40]
       mov      rbx, rdi
       xor      r15d, r15d
       mov      r14d, dword ptr [rbx+0x08]
       test     r14d, r14d
       jle      G_M41176_IG10
G_M41176_IG03:  ;; offset=0x0026
       mov      edi, r15d
       mov      edi, dword ptr [rbx+4*rdi+0x10]
       call     [System.Number:Int32ToDecStr(int):System.String]
       mov      r13, rax
       test     r13, r13
       je       SHORT G_M41176_IG07
       mov      r12d, dword ptr [r13+0x08]
       test     r12d, r12d
       je       SHORT G_M41176_IG07
       mov      eax, r12d
       mov      dword ptr [rbp-0x2C], eax
       lea      edi, [rax+0x01]
       test     edi, edi
       jl       G_M41176_IG12
       jmp      SHORT G_M41176_IG08
G_M41176_IG07:  ;; offset=0x0057
       mov      rdi, 0x76E7440F8688      ; ' '
       jmp      SHORT G_M41176_IG09
G_M41176_IG08:  ;; offset=0x0063
       call     System.String:FastAllocateString(int):System.String
       mov      rdi, rax
```

```
        mov      gword ptr [rbp-0x38], rdi
        cmp      byte  ptr [rdi], dil
        lea      rsi, bword ptr [rdi+0x0C]
        add      r13, 12
        mov      edx, r12d
        add      rdx, rdx
        mov      rdi, rsi
        mov      rsi, r13
        call     [System.Buffer:Memmove(byref,byref,ulong)]
        movsxd   rax, dword ptr [rbp-0x2C]
        mov      r13, gword ptr [rbp-0x38]
        lea      rax, bword ptr [r13+2*rax+0x0C]
        mov      rcx, 0x76E7440F8694
        movzx    rdx, word  ptr [rcx]
        mov      word  ptr [rax], dx
        mov      rdi, r13
G_M41176_IG09:  ;; offset=0x00AC
        call     [System.Console:Write(System.String)]
        inc      r15d
        cmp      r14d, r15d
        jg       G_M41176_IG03
G_M41176_IG10:  ;; offset=0x00BE
        call     [System.Console:WriteLine()]
        nop
        add      rsp, 24
        pop      rbx
        pop      r12
        pop      r13
        pop      r14
        pop      r15
        pop      rbp
        ret
G_M41176_IG12:  ;; offset=0x00D4
        call     [System.ThrowHelper:ThrowOutOfMemoryException_StringTooLong()]
        int3
```

Сортування бульбашкою мовою Java

```
class BubbleSort {
   public static void sort(int[] arr) {
      int n = arr.length;
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
```

```
                // Swap elements
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        System.out.println("Unsorted array:");
        printArray(arr);

        sort(arr);

        System.out.println("\nSorted array:");
        printArray(arr);
    }

    private static void printArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Сортування бульбашкою у байт-коді віртуальною машиною JDK 21.0.2

```
class BubbleSort {
  BubbleSort();
     0: aload_0
     1: invokespecial #1                  // Method java/lang/Object."<init>":()V
     4: return


  public static void sort(int[]);
     0: aload_0
     1: arraylength
     2: istore_1
     3: iconst_0
     4: istore_2
     5: iload_2
```

```
 6: iload_1
 7: iconst_1
 8: isub
 9: if_icmpge     66
12: iconst_0
13: istore_3
14: iload_3
15: iload_1
16: iload_2
17: isub
18: iconst_1
19: isub
20: if_icmpge     60
23: aload_0
24: iload_3
25: iaload
26: aload_0
27: iload_3
28: iconst_1
29: iadd
30: iaload
31: if_icmple     54
34: aload_0
35: iload_3
36: iaload
37: istore        4
39: aload_0
40: iload_3
41: aload_0
42: iload_3
43: iconst_1
44: iadd
45: iaload
46: iastore
47: aload_0
48: iload_3
49: iconst_1
50: iadd
51: iload         4
53: iastore
54: iinc          3, 1
57: goto          14
```

```
   60: iinc          2, 1
   63: goto          5
   66: return


public static void main(java.lang.String[]);
    0: bipush        7
    2: newarray       int
    4: dup
    5: iconst_0
    6: bipush        64
    8: iastore
    9: dup
   10: iconst_1
   11: bipush        34
   13: iastore
   14: dup
   15: iconst_2
   16: bipush        25
   18: iastore
   19: dup
   20: iconst_3
   21: bipush        12
   23: iastore
   24: dup
   25: iconst_4
   26: bipush        22
   28: iastore
   29: dup
   30: iconst_5
   31: bipush        11
   33: iastore
   34: dup
   35: bipush        6
   37: bipush        90
   39: iastore
   40: astore_1
   41: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
   44: ldc           #13                 // String Unsorted array:
   46: invokevirtual #15                 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
   49: aload_1
   50: invokestatic  #21                 // Method printArray:([I)V
```

```
      53: aload_1
      54: invokestatic  #27                 // Method sort:([I)V
      57: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
      60: ldc           #30                 // String \nSorted array:
      62: invokevirtual #15                 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      65: aload_1
      66: invokestatic  #21                 // Method printArray:([I)V
      69: return


  private static void printArray(int[]);
      0: aload_0
      1: astore_1
      2: aload_1
      3: arraylength
      4: istore_2
      5: iconst_0
      6: istore_3
      7: iload_3
      8: iload_2
      9: if_icmpge     36
     12: aload_1
     13: iload_3
     14: iaload
     15: istore        4
     17: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
     20: iload         4
     22: invokedynamic #32,  0             // InvokeDynamic
#0:makeConcatWithConstants:(I)Ljava/lang/String;
     27: invokevirtual #36                 // Method java/io/PrintStream.print:(Ljava/lang/String;)V
     30: iinc          3, 1
     33: goto          7
     36: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
     39: invokevirtual #39                 // Method java/io/PrintStream.println:()V
     42: return

  }
```