Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

## Кафедра прикладной математики и кибернетики

Современные технологии программирования

Практическая работа №12

«Вычисление метрических характеристик реализаций алгоритмов»

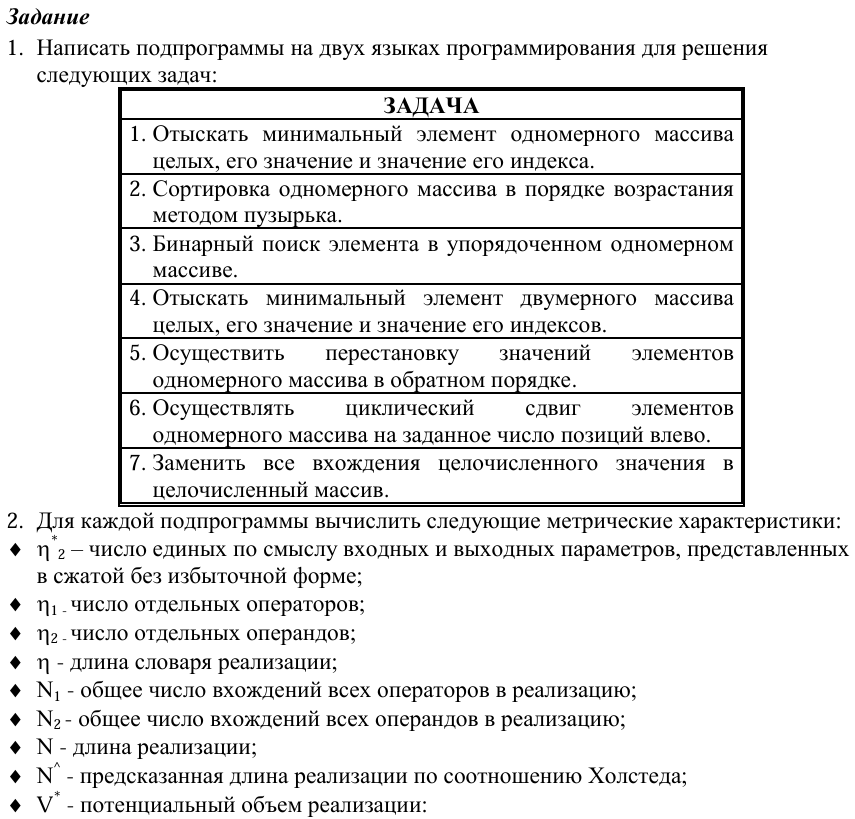
Выполнил: студент 4 курса

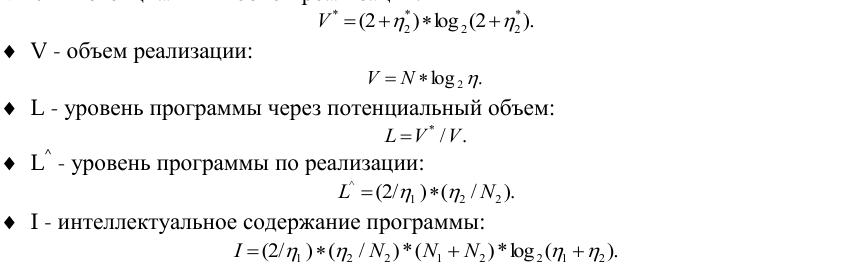
группы ИП-111 Кузьменок Денис Витальевич

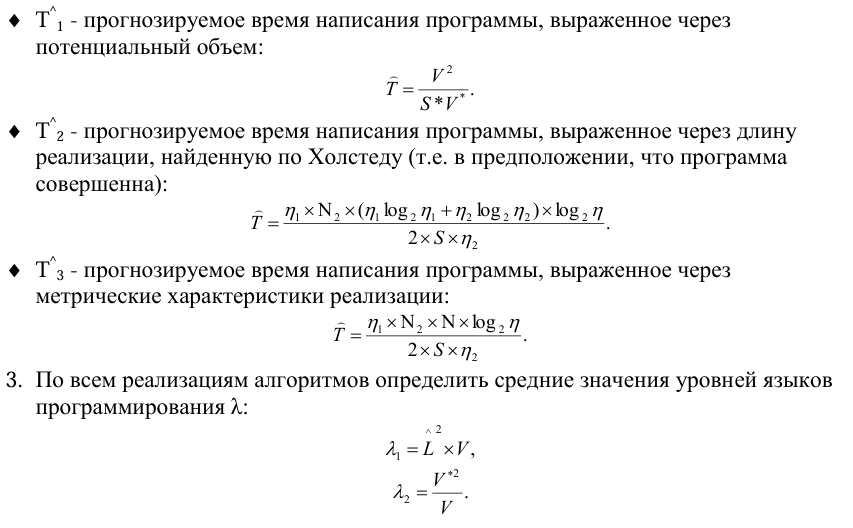
Проверил преподаватель: Зайцев Михаил Георгиевич

Новосибирск, 2024 г.

**Задание**

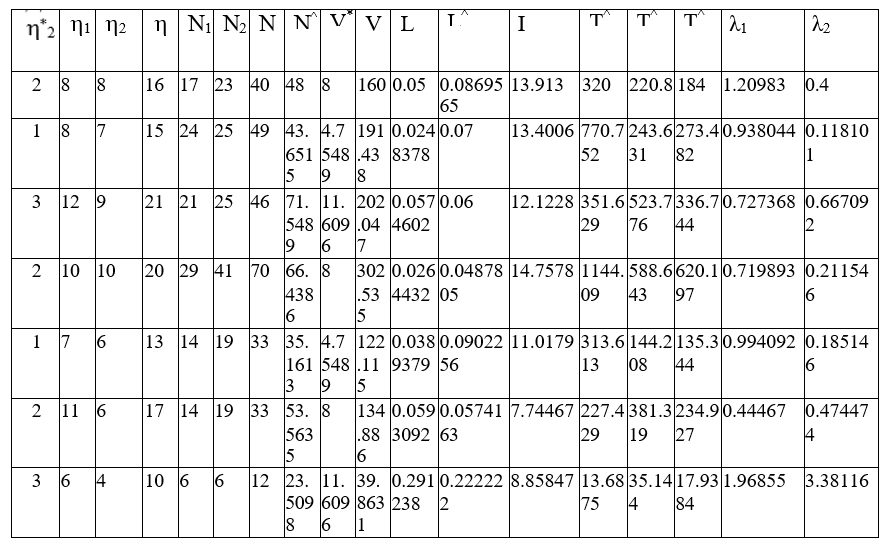




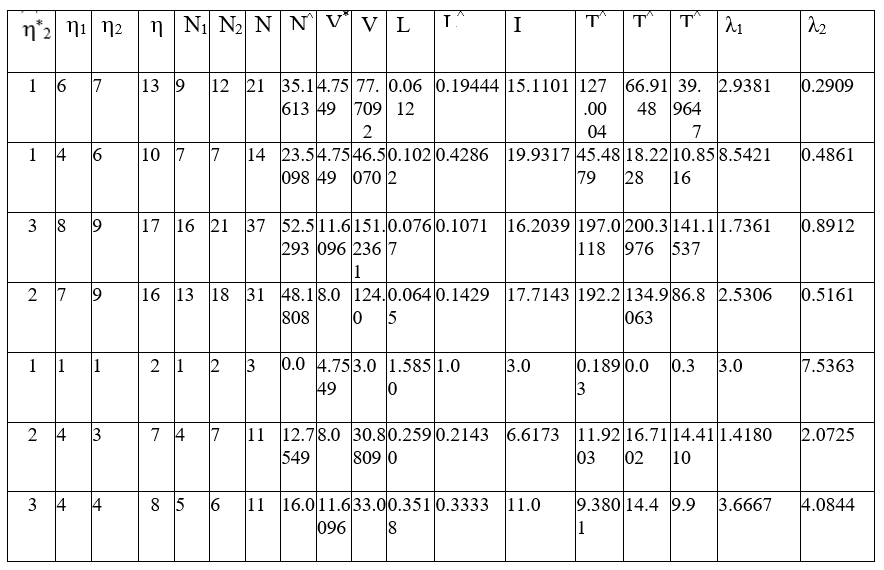


**Ход работы**

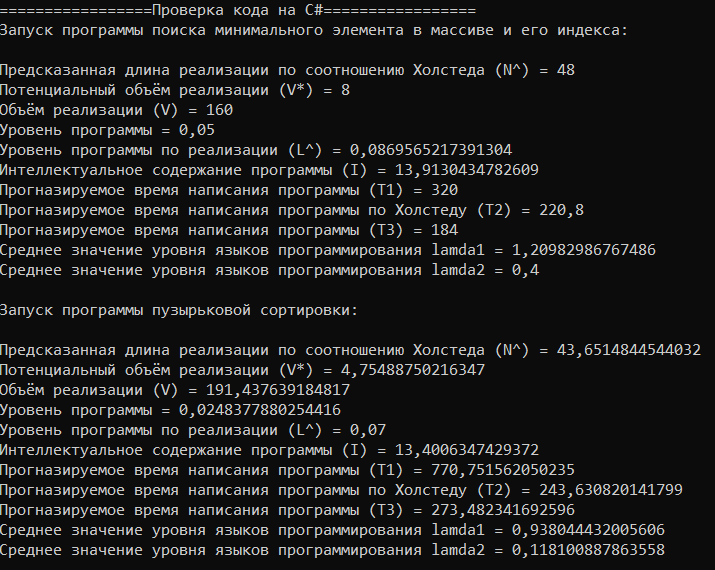
Программа на языке C# показала следующие результаты:

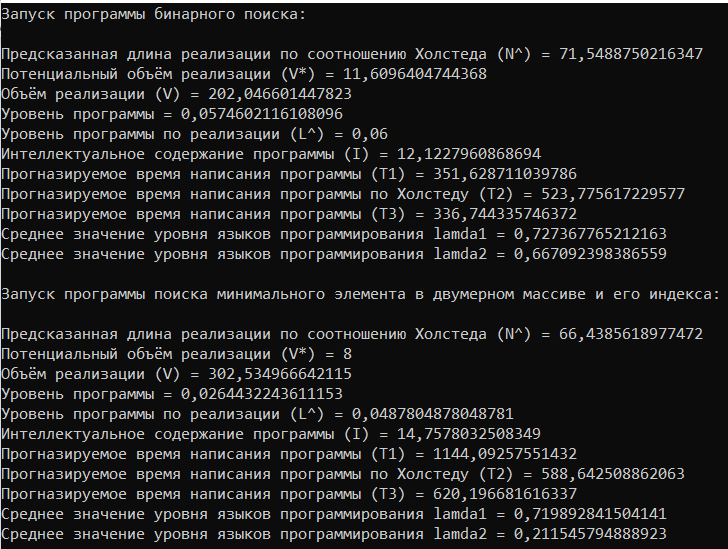


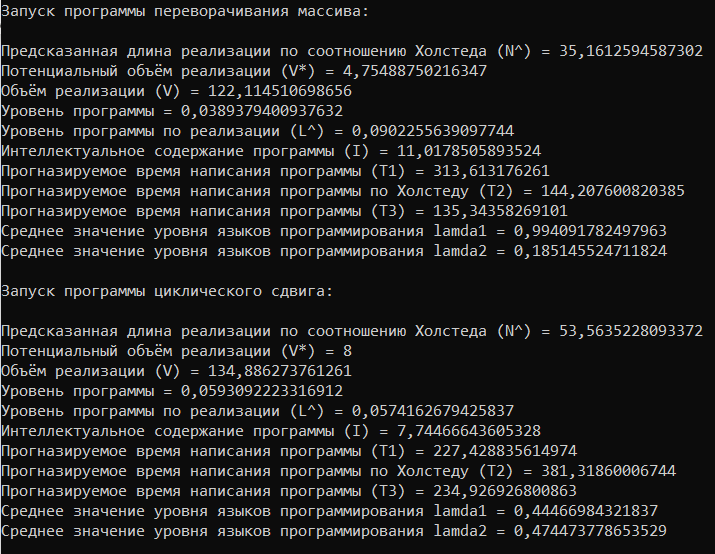
Программа на языке Python показала следующие результаты:

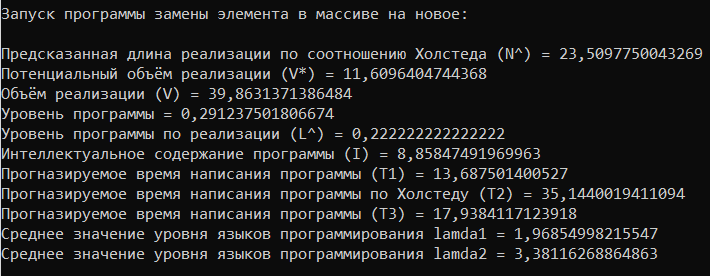


Вывод результатов программы на C#:

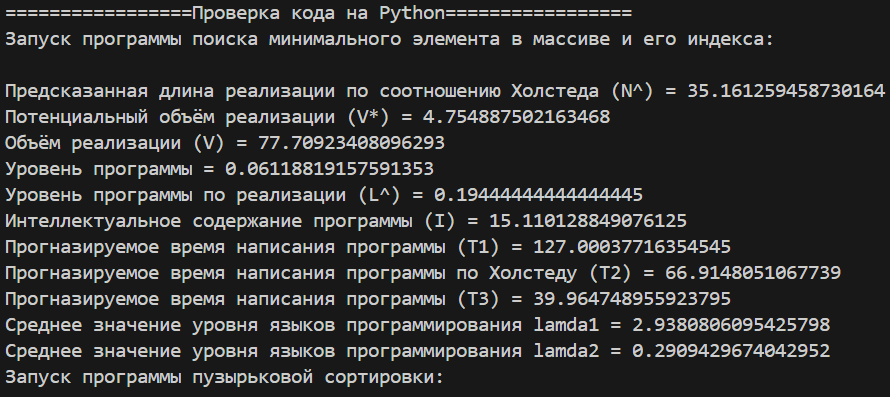


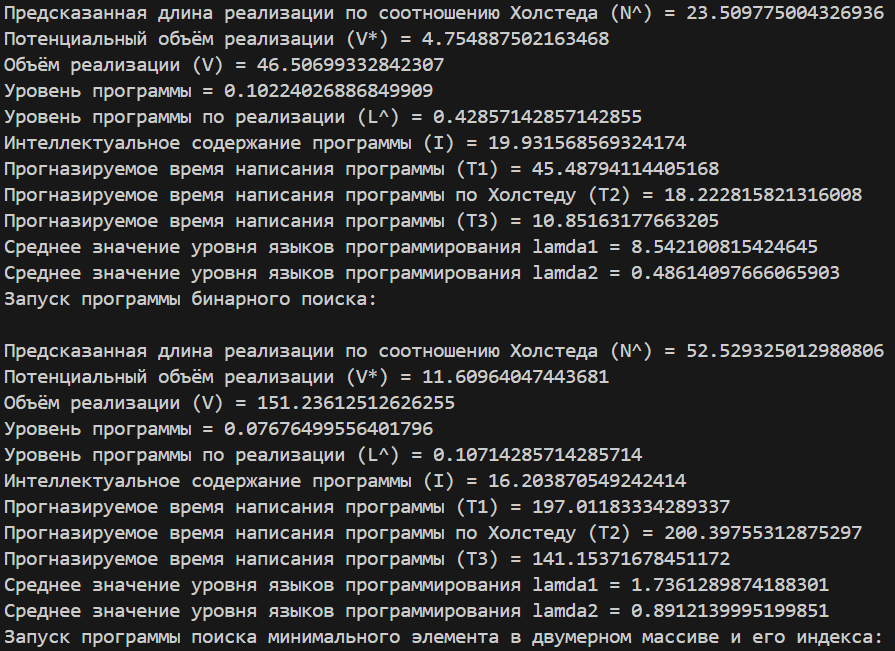


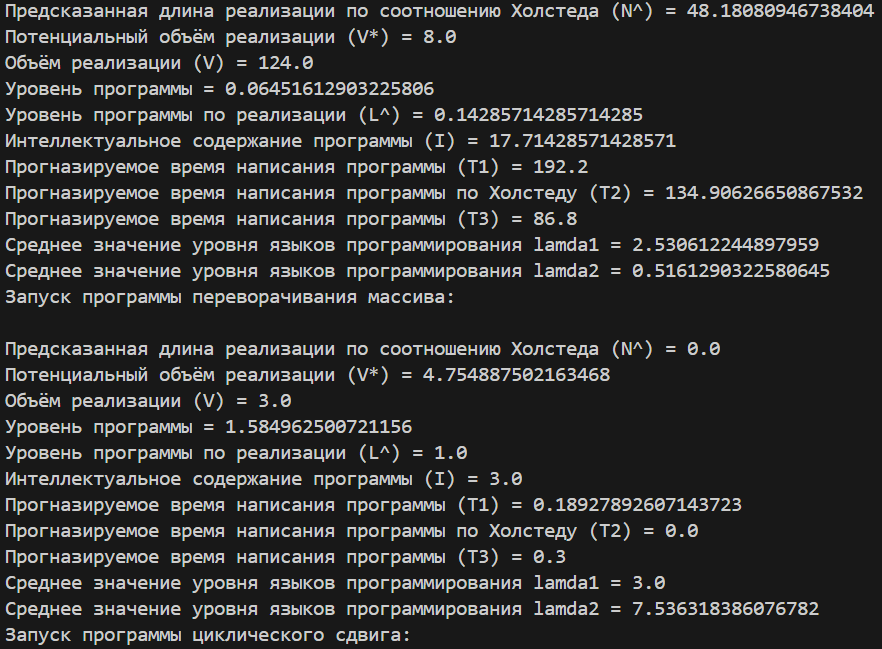


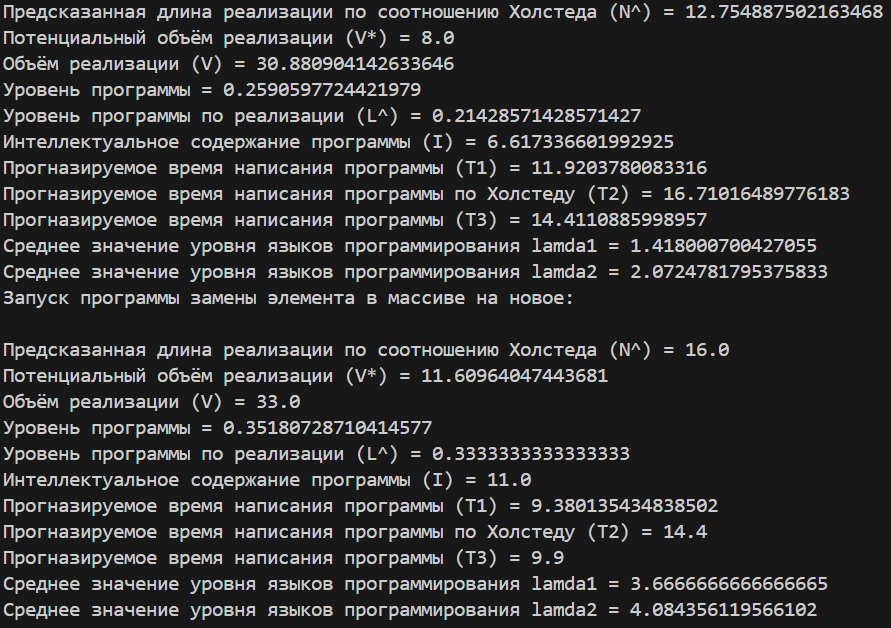


Вывод программы на Python:









**Вывод:**

Лабораторная работа позволила глубже понять процесс разработки

программных модулей на разных языках программирования, а также оценить их сложность и прогнозируемое время разработки с помощью метрик

Холстеда. Методы оценки сложности программного кода являются полезными инструментами для оценки эффективности программного

обеспечения, а также для планирования времени разработки и оптимизации кода.

**Листинг программ**

**Program.cs**

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Linq**;

**using** **System.Runtime.Remoting.Messaging**;

**using** **System.Text**;

**using** **System.Threading.Tasks**;

**namespace** **lab12**

{

**class** **Program**

{

**private** **static** **void** **Main**(**string**[] args)

{

Console.WriteLine("=================Проверка кода на C#=================");

Console.WriteLine("Запуск программы поиска минимального элемента в массиве и его индекса:\n");

Calculate(**2**, **8**, **8**, **17**, **23**);

Console.WriteLine("Запуск программы пузырьковой сортировки:\n");

Calculate(**1**, **8**, **7**, **24**, **25**);

Console.WriteLine("Запуск программы бинарного поиска:\n");

Calculate(**3**, **12**, **9**, **21**, **25**);

Console.WriteLine("Запуск программы поиска минимального элемента в двумерном массиве и его индекса:\n");

Calculate(**2**, **10**, **10**, **29**, **41**);

Console.WriteLine("Запуск программы переворачивания массива:\n");

Calculate(**1**, **7**, **6**, **14**, **19**);

Console.WriteLine("Запуск программы циклического сдвига:\n");

Calculate(**2**, **11**, **6**, **14**, **19**);

Console.WriteLine("Запуск программы замены элемента в массиве на новое:\n");

Calculate(**3**, **6**, **4**, **6**, **6**);

}

**private** **static** **void** **Calculate**(**double** udot, **double** u1, **double** u2, **double** N1, **double** N2)

{

**int** S = **10**;

**double** V = (N1 + N2) \* Math.Log((u1 + u2), **2**);

**double** Vdot = (**2** + udot) \* Math.Log((**2** + udot), **2**);

**double** Lgalka = (**2** / u1) \* (u2 / N2);

Console.WriteLine($"Предсказанная длина реализации по соотношению Холстеда (N^) = {u1 \* Math.Log(u1, 2) + u2 \* Math.Log(u2, 2)}");

Console.WriteLine($"Потенциальный объём реализации (V\*) = {Vdot}");

Console.WriteLine($"Объём реализации (V) = {V}");

Console.WriteLine($"Уровень программы = {Vdot / V}");

Console.WriteLine($"Уровень программы по реализации (L^) = {Lgalka}");

Console.WriteLine($"Интеллектуальное содержание программы (I) = {(2 / u1) \* (u2 / N2) \* (N1 + N2) \* Math.Log((u1 + u2), 2)}");

Console.WriteLine($"Прогназируемое время написания программы (T1) = {(V \* V / (S \* Vdot))}");

Console.WriteLine($"Прогназируемое время написания программы по Холстеду (T2) = {(u1 \* N2 \* (u1 \* Math.Log(u1, 2) + u2 \* Math.Log(u2, 2)) \* Math.Log((u1 + u2), 2)) / (2 \* S \* u2)}");

Console.WriteLine($"Прогназируемое время написания программы (T3) = {(u1 \* N2 \* (N1 + N2) \* Math.Log((u1 + u2), 2)) / (2 \* S \* u2)}");

Console.WriteLine($"Среднее значение уровня языков программирования lamda1 = {Lgalka \* Lgalka \* V}");

Console.WriteLine($"Среднее значение уровня языков программирования lamda2 = {Vdot \* Vdot / V}\n");

}

**private** **static** (**int** , **int**) FindMin(List<**int**> a)

{

**if** (a.Count == **0**) **return** (**int**.MaxValue, -**1**);

**int** minVal = a[**0**];

**int** minIndex = **0**;

**for**(**int** i = **1**; i < a.Count; i++)

{

**if** (a[i] < minVal)

{

minVal = a[i];

minIndex = i;

}

}

**return** (minVal, minIndex);

}

**private** **static** **void** **BubbleSort**(List<**int**> a)

{

**for**(**int** i = **0**; i < a.Count - **1**; i++)

{

**for**(**int** j = a.Count - **1**; j > i; j--)

{

**if** (a[j] < a[j - **1**])

{

**int** temp = a[j];

a[j] = a[j - **1**];

a[j - **1**] = temp;

}

}

}

}

**private** **static** **int** **BinarySearch**(List<**int**> a, **int** target)

{

**int** left = **0**, right = a.Count - **1**;

**while**(left <= right)

{

**int** mid = (left + right) / **2**;

**if** (a[mid] == target) **return** mid;

**if** (a[mid] < target) left = mid + **1**;

**else** right = mid - **1**;

}

**return** -**1**;

}

**private** **static** (**int**, **int**, **int**) FindMinMatrix(List<List<**int**>> a)

{

**if** (a.Count == **0** || a[**0**].Count == **0**) **return** (**int**.MaxValue, -**1**, **1**);

**int** minVal = a[**0**][**0**], minRow = **0**, minCol = **0**;

**for**(**int** i = **0**; i < a.Count; i++)

{

**for**(**int** j = **0**; j < a[i].Count; j++)

{

**if** (a[i][j] < minVal)

{

minVal = a[i][j];

minRow = i;

minCol = j;

}

}

}

**return** (minVal, minRow, minCol);

}

**private** **static** **void** **Reverse**(List<**int**> a)

{

**int** left = **0**, right = a.Count - **1**;

**while**(left < right)

{

**int** temp = a[left];

a[left] = a[right];

a[right] = temp;

left++;

right--;

}

}

**private** **static** **void** **CycleShift**(List<**int**> a, **int** count)

{

count %= a.Count;

**if**(count == **0**) **return**;

List<**int**> temp = **new** List<**int**>(a);

**for**(**int** i = **0**; i < a.Count; i++)

{

a[i] = temp[(i + count) % a.Count];

}

}

**private** **static** **void** **ReplaceValue**(List<**int**> a, **int** oldValue, **int** newValue)

{

**for**(**int** i = **0**; i < a.Count; i++)

{

**if** (a[i] == oldValue)

{

a[i] = newValue;

}

}

}

}

}

**Main.py**

**import** **math**

**def** **calculate**(udot, u1, u2, N1, N2):

S = **10**

V = (N1 + N2) \* math.log2(u1 + u2)

Vdot = (**2** + udot) \* math.log2(**2** + udot)

Lgalka = (**2** / u1) \* (u2 / N2)

print(f"Предсказанная длина реализации по соотношению Холстеда (N^) = {u1 \* math.log2(u1) + u2 \* math.log2(u2)}")

print(f"Потенциальный объём реализации (V\*) = {Vdot}")

print(f"Объём реализации (V) = {V}")

print(f"Уровень программы = {Vdot / V}")

print(f"Уровень программы по реализации (L^) = {Lgalka}")

print(f"Интеллектуальное содержание программы (I) = {(2 / u1) \* (u2 / N2) \* (N1 + N2) \* math.log2(u1 + u2)}")

print(f"Прогназируемое время написания программы (T1) = {V \* V / (S \* Vdot)}")

print(f"Прогназируемое время написания программы по Холстеду (T2) = {(u1 \* N2 \* (u1 \* math.log2(u1) + u2 \* math.log2(u2)) \* math.log2(u1 + u2)) / (2 \* S \* u2)}")

print(f"Прогназируемое время написания программы (T3) = {(u1 \* N2 \* (N1 + N2) \* math.log2(u1 + u2)) / (2 \* S \* u2)}")

print(f"Среднее значение уровня языков программирования lamda1 = {Lgalka \* Lgalka \* V}")

print(f"Среднее значение уровня языков программирования lamda2 = {Vdot \* Vdot / V}")

**def** **find\_min**(a):

**if** **not** a:

**return** (float('inf'), -**1**)

min\_val = a[**0**]

min\_index = **0**

**for** i **in** range(**1**, len(a)):

**if** a[i] < min\_val:

min\_val = a[i]

min\_index = i

**return** (min\_val, min\_index)

**def** **bubble\_sort**(a):

n = len(a)

**for** i **in** range(n-**1**):

**for** j **in** range(n-i-**1**):

**if** a[j] > a[j+**1**]:

a[j], a[j+**1**] = a[j+**1**], a[j]

**def** **binary\_search**(a, target):

left, right = **0**, len(a) - **1**

**while** left <= right:

mid = (left + right) // **2**

**if** a[mid] == target:

**return** mid

**elif** a[mid] < target:

left = mid + **1**

**else**:

right = mid - **1**

**return** -**1**

**def** **find\_min\_matrix**(a):

**if** **not** a **or** **not** a[**0**]:

**return** (float('inf'), -**1**, -**1**)

min\_val = a[**0**][**0**]

min\_row, min\_col = **0**, **0**

**for** i **in** range(len(a)):

**for** j **in** range(len(a[i])):

**if** a[i][j] < min\_val:

min\_val = a[i][j]

min\_row, min\_col = i, j

**return** (min\_val, min\_row, min\_col)

**def** **reverse**(a):

a.reverse()

**def** **cycle\_shift**(a, positions):

positions %= len(a)

**if** positions == **0**:

**return**

a[:] = a[-positions:] + a[:-positions]

**def** **replace\_value**(a, old\_value, new\_value):

**for** i **in** range(len(a)):

**if** a[i] == old\_value:

a[i] = new\_value

**if** \_\_name\_\_ == "\_\_main\_\_":

print("=================Проверка кода на Python=================")

print("Запуск программы поиска минимального элемента в массиве и его индекса:**\n**")

calculate(**1**, **6**, **7**, **9**, **12**)

print("Запуск программы пузырьковой сортировки:**\n**")

calculate(**1**, **4**, **6**, **7**, **7**)

print("Запуск программы бинарного поиска:**\n**")

calculate(**3**, **8**, **9**, **16**, **21**)

print("Запуск программы поиска минимального элемента в двумерном массиве и его индекса:**\n**")

calculate(**2**, **7**, **9**, **13**, **18**)

print("Запуск программы переворачивания массива:**\n**")

calculate(**1**, **1**, **1**, **1**, **2**)

print("Запуск программы циклического сдвига:**\n**")

calculate(**2**, **4**, **3**, **4**, **7**)

print("Запуск программы замены элемента в массиве на новое:**\n**")

calculate(**3**, **4**, **4**, **5**, **6**)