Міністерство освіти і науки України

Львівський національний університет ім. І.Я.Франка

**Звіт**

# Навчальна практика

# (python, C++)

Виконав:

Студент групи ПМп-11

Грищишин Роман

**Львів 2017**

Зміст

Завдання №1……………………………………………

Завдання №2……………………………………………

Завдання №3……………………………………………

Завдання №4……………………………………………

Завдання №5……………………………………………

Завдання №6……………………………………………

Завдання №7……………………………………………

Завдання №8……………………………………………

Завдання №9……………………………………………

Завдання №1

Умова:

### Для заданого n обчислити суму:

Короткий опис роботи програми:

### Виконуючи два цикли програма обраховує і сумує добутки.

Сирцевий код:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | n = int(input('n= '))  summa = 0  multiplication = 1  for i in range(1, (n + 1)):  for k in range(1, (i\*2+1)):  multiplication \*= (k+1)  summa += multiplication  multiplication = 1    print("Сумма дорівнює", summa) | |

Приклад виконання:

### Input:

|  |
| --- |
| n= 10 |

Output:

|  |
| --- |
| Сумма дорівнює 51212944273488041646 |

Завдання №2

Умова:

### Використовуючи алгоритм Евкліда знайти найбільший спільний дільник двох чисел.

Короткий опис роботи програми:

### Програма рекурсивно шукає спільний дільник.

Сирцевий код:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13 | def nsd(n, m):  if m == 0:  return n  elif n == 0:  return m  elif m > n:  return nsd(n, m % n)  else:  return nsd(n % m, m)  print(nsd(int(input("1'st num ")), int(input("2'nd num ")))) .    print("Сумма дорівнює", summa) | |

Приклад виконання:

### Input:

|  |
| --- |
| 1'st num 16  2'nd num 32 |

Output:

|  |
| --- |
| 16 |

Завдання №3

Умова:

### Прочитати дійсну матрицю nxn (n > 5), вияснити чи задана матриця є симетричною відносно головної діагоналі. Якщо ні – перевірити чи перестановкою рядків можна досягти заданої властивості.

Короткий опис роботи програми:

### Спершу, функція порівнює елементи на i j та j i позиціях, якщо вони відрізняються, то матриця не є симетричною. Далі, ми генеруємо всі перестановки рядків цієї матриці і виконуємо з ними ті самі дії.

Сирцевий код:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | import itertools  def output\_matrix(matrix):  for row in matrix:  for elem in row:  print(elem, end = " ")  else:  print()  def input\_matrix():  matrix = []  n = int(input("Введіть розмір матриці: "))  for m in range(n):  matrix.append([])  for row in range(len(matrix)):  for k in range(n):  print("Елемент {0}-го рядку, {1}-го стовпчика".format(row + 1, k + 1), end = " ")  matrix[row].append(int(input()))  return matrix  def perevirka(matrix):  for x in range(len(matrix)):  for m in range(len(matrix)):  if not(matrix[x][m] == matrix[m][x]):  return False  else:  return True  mat = input\_matrix()  print("Ваша матриця")  output\_matrix(mat)  if perevirka(mat):  print("Матриця симетрична відносно головної діагоналі")  else:  permutations = list(itertools.permutations(mat))  for mat in permutations:  if perevirka(mat):  print("Симетрії матриці відносно голоної діагоналі можна досягнути перестановкою рядків")  break  else:  print("Симетрії матриці відносно голоної діагоналі не можна досягнути перестановкою рядків") | |

Приклад виконання:

### Input:

|  |
| --- |
| Введіть розмір матриці: 3  Елемент 1-го рядку, 1-го стовпчика 3  Елемент 1-го рядку, 2-го стовпчика 3  Елемент 1-го рядку, 3-го стовпчика 3  Елемент 2-го рядку, 1-го стовпчика 2  Елемент 2-го рядку, 2-го стовпчика 3  Елемент 2-го рядку, 3-го стовпчика 3  Елемент 3-го рядку, 1-го стовпчика 2  Елемент 3-го рядку, 2-го стовпчика 3  Елемент 3-го рядку, 3-го стовпчика 2 |

Output:

|  |
| --- |
| Ваша матриця  3 3 3  2 3 3  2 3 2  Симетрії матриці відносно голоної діагоналі можна досягнути перестановкою рядків |

Завдання №4

Умова:

### Прочитати текстову стрічку довжиною не більше ніж 253 символи. Вважається, що у стрічку входять слова, які розділені між собою хоча б одним пропуском. Надрукувати слова, які в заданому тексті є найдовшими і найкоротшими.

Короткий опис роботи програми:

### Програма розбиває введену стрічку на список слів. Потім, порівнюючи довжини цих слів програма знаходить найдовше та найкоротше слово. Слова, довжини яких дорівнюють довжинам найдовшого та найкоротшого слова записуємо в списки з найдовшими та найкоротшими словами відповідно. Виводимо ці слова у консоль.

Сирцевий код:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | string = input().split()  longest = string[0]  shortest = string[0]  for word in string:  if len(word) > len(longest):  longest = word  if len(word) < len(shortest):  shortest = word  shortest\_list = []  longest\_list = []  for word in string:  if len(word) == len(shortest):  shortest\_list.append(word)  if len(word) == len(longest):  longest\_list.append(word)  print("The longest words are: ", longest\_list)  print("The shortest words are: ", shortest\_list) |

Приклад виконання:

### Input:

|  |
| --- |
| Test string to find the longest and the shortest words in string |

Output:

|  |
| --- |
| The longest words are: ['shortest']  The shortest words are: ['to', 'in'] |

Завдання №5

Умова:

### Прочитати m груп (2 ≤ m ≤ 10) по n штук в групі (4 ≤ n ≤ 10) трицифрових цілих додатніх чисел. Надрукувати ту групу, яка містить найбільшу кількість простих чисел. Написати підпрограми для розпізнавання

Короткий опис роботи програми:

### Спершу користувач вводить кількість і розмір груп. Щоб зекономити час програма генерує трицифрові числа самостійно. Потім, використовуючи тест на простоту (перевіряє, чи число ділиться на числа менші за себе) ми знаходимо кількість простих чисел в кожній групі. Вибираємо групу, в якій найбільше простих чисел і виводимо її.

Сирцевий код:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | from random import randrange  def input\_group(n, m):  groups = []  for x in range(m):  groups.append([])  for group in groups:  for x in range(n):  group.append(randrange(100, 1000))  return groups  def output(groups\_func):  counter = 1  for group in groups\_func:  print("Group {} contains:".format(counter))  counter += 1  for x in group:  print(x, end = " ")  else:  print()  def primarity\_test(number):  for x in range(2, number):  if number % x == 0:  return False  else:  return True  n = int(input("n = "))  m = int(input("m = "))  my\_groups = input\_group(n, m)  output(my\_groups)  biggest\_group\_counter = 0  for group in my\_groups:  counter = 0  for number in group:  if primarity\_test(number):  counter += 1  else:  if biggest\_group\_counter < counter:  biggest\_group = group  biggest\_group\_counter = counter  print("""There are the most primal numbers in this group:{}  There are {} primal numbers in it""".format(biggest\_group, biggest\_group\_counter)  ) | |

Приклад виконання:

### Input:

|  |
| --- |
| n = 3  m = 3 |

Output:

|  |
| --- |
| Group 1 contains:  439 536 135  Group 2 contains:  781 843 357  Group 3 contains:  794 206 345  There are the most primal numbers in this group:[439, 536, 135]  There are 1 primal numbers in it |

Завдання №4

Умова:

### C:\Users\Red Sun\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Снимок.pngОписати рекурсивну функцію, яка б використовуючи схему Горнера обчислювала значення многочлена

в заданій точці х0.

Короткий опис роботи програми:

### Функція описана таким чином: приймаємо вектор зі значеннями коефіцієнтів при змінних в поліномі та значення х. Передаємо змінній а значення останнього елемента вектора і видаляємо цей елемент з вектора. Якщо вектор пустий: повертаємо а, в іншому випадку: повертаємо значення змінної х, помноженої на повторний виклик функції та додаємо а.

### Сама програма працює таким чином. Користувач вводить х та коефіцієнти полінома за зростанням степеня змінної. Значення записуються у вектор. Потім вектор розвертається і викликається функція. Виводимо значення функції.

Сирцевий код:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | #include <iostream>  #include <vector>  using namespace std;  double Horners\_method(vector <float> coefficients, float x\_value) {  double a = coefficients.back();  coefficients.pop\_back();  if (coefficients.empty())  return a;  else  return x\_value \* Horners\_method(coefficients, x\_value) + a;  }  int main()  {  cout << "Enter size of polynomial: " << endl;  short size; cin >> size;  vector <float> polynomial;  cout << "Enter coefficients in this format: A1 + A2x + A3x^2... AnX^n (terminated by \"s\")" << endl;  double temp;  for (int i = 1; i <= size; i++) {  cout << "Enter A" << i  << " coefficient : ";  cin >> temp;  polynomial.push\_back(temp);  }  reverse(polynomial.begin(), polynomial.end());      cout << "Enter x " << endl;    float x; cin >> x;  if (!cin) cerr << "incorrect value";    cout << Horners\_method(polynomial, x) << endl;    system("pause");  } |

Приклад виконання:

### Input:

|  |
| --- |
| Enter size of polynomial:  5  Enter coefficients in this format: A1 + A2x + A3x^2... AnX^n (terminated by "s")  Enter A1 coefficient : 1  Enter A2 coefficient : 2  Enter A3 coefficient : 3  Enter A4 coefficient : 4  Enter A5 coefficient : 5  Enter x 2 |

Output:

|  |
| --- |
| 129 |