**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Спеціальність програмна інженерія**

***Алгоритми та складність***

**Завдання № 2**

**Звіт**

**Виконала:**

студент групи К-28

Cаллям Салех

**Київ-2019**

*Умова завдання:*

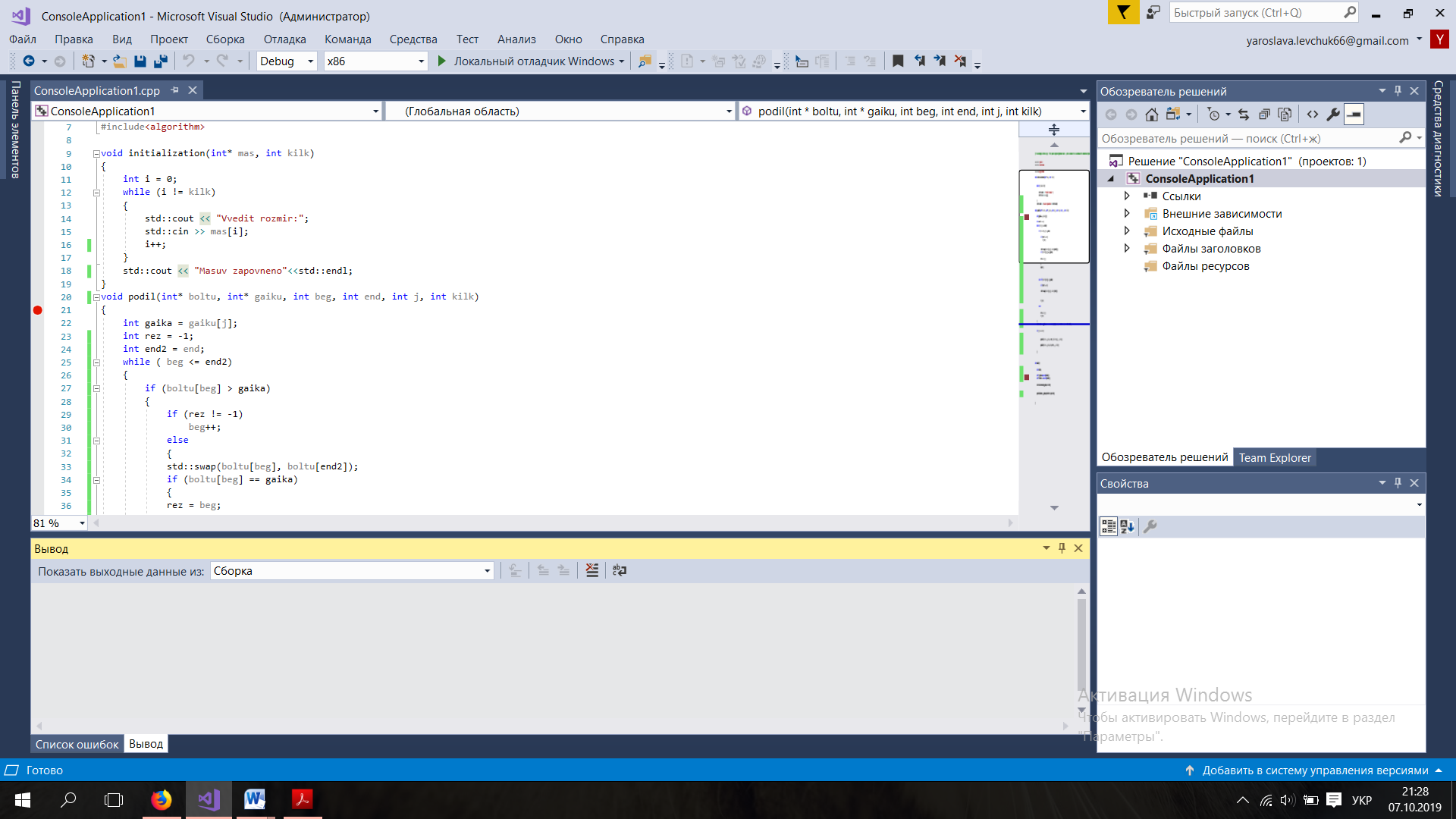
Нехай є n болтів різного розміру та n відповідних гайок. Припустимо, можна порівнювати, чи підходять гайка і болт одне до одного, або гайка більша (чи менша). Порівняти між собою дві гай-ки чи два болти неможливо. Розробіть і реалізуйте алгоритм розбивки всіх гайок і болтів на відповідні пари за час Θ(n log n).

*Опис коду програми:*

*Ініціалізація:*

1. Вказуємо к-сть пар гайок та болтів;
2. Далі створюємо і ініціалізуємо два масиви: 1-дані про розмір гайок, 2- розмір болтів.

*Функція для ініціалізації:*



*Опишемо алгоритм сортування:* Сортування та знаходження пар відбувається у рекурсивній функції. Оскільки дана функція має тип void , то нічого не повертає.

*Аргументи* void podil(int\* boltu, int\* gaiku, int beg, int end, int j, int kilk):

1.int\* gaiku- масив заданих гайок;

2.int\* boltu- масив заданих болтів;

3. int end – індекс кінцевого елемента;

4.int beg – індекс початкового елемента, з якого починається перевірка;

5.int kilk- кількість гайок і болтів;

6.int j- індекс гайки, відносно якої відбувається сортування та пошук пари;

*Сортування:*

1.фіксуємо одну гайку;

2.перевіряємо розміри гайки відносно болтів у циклі, цикл буде виконуватись поки початок не стане більшим кінця масиву);

3.міняємо місцями поточний і кінцевий болти, якщо гайка менша болта і до цього пара не була знайдена, при цьому зменшуємо розмір масиву на один з кінця;

4.якщо гайка менша болта і до цього пара була знайдена, то переходимо до наступного болта;

5.якщо гайка більша болта і до цього пара не була знайдена, то переходимо до наступного болта;

6.якщо гайка більша болта і до цього пара була знайдена, то міняємо місцями поточний болт і «болт-пару». При цьому змінюємо індекс результуючого болта на поточний і переходимо до наступного болта;

7.якщо гайка підходить до болта, то запам’ятовуємо індекс і переходимо до наступного болта;

8.коли вийдемо з циклу, то порівнюємо наступну гайку з розмірами болта, який підходив до попередньої гайки. В залежності від розміру передаємо відповідповідні аргументи, якщо більший, то працюємо з масивом елементів до знайденого болта, інакше з масивом після знайденого болта.

*Завершення:*

Програма завершується коли пройдені всі болти і для кожного знайдена пара.

*Вивід:*

На екран виводяться пари за розміром та в дужках вказані їхні індекси в загальному відсортованому масиві.

*Аналіз алгоритму:*

Середній час роботи алгоритму Θ(n log n)

Найгірший час роботи Θ(n^2)

*Реалізація алгоритму:*

Реалізовано на С++