Кожевніков Ілля КН-923в
Лабораторна робота №7.
Функції.

Скопіював та модифікував лабораторну 23. (**) Перетворити число (максимальне значення якого - 9999) в рядок. (усі символи нижнього регістру на виході).
Наприклад,
• 123 – “one hundred twenty three”,
• 4311 – “four thousands three hundreds eleven

Обав додаткове завдання 1. Реалізувати функцію (та продемонструвати її роботу), що визначає, скільки серед заданої послідовності чисел таких пар, у котрих перше число менше наступного, використовуючи функцію з варіативною кількістю аргументів.

Результат

```
kozhevka@AbobaLaptop:~/programming-kozhevnikov/lab07/build$ ./main.bin
Generated random number: 174081842
1 < 7; 0 < 8; 1 < 8;
Current less next iteration count: 3
Randomly generated number 1842.
Write as: one thousand eight hundred forty two
kozhevka@AbobaLaptop:~/programming-kozhevnikov/lab07/build$ ./main.bin
Generated random number: 2019298149
0 < 1; 1 < 9; 2 < 9; 1 < 4; 4 < 9;
Current less next iteration count: 5
Randomly generated number 8149.
Write as: eight thousand one hundred forty nine
kozhevka@AbobaLaptop:~/programming-kozhevnikov/lab07/build$ ./main.bin
Generated random number: 2019298149
0 < 1; 1 < 9; 2 < 9; 1 < 4; 4 < 9;
Current less next iteration count: 5
Randomly generated number 8149.
Write as: eight thousand one hundred forty nine
kozhevka@AbobaLaptop:~/programming-kozhevnikov/lab07/build$ ./main.bin
Generated random number: 2019298149
0 < 1; 1 < 9; 2 < 9; 1 < 4; 4 < 9;
Current less next iteration count: 5
Randomly generated number 8149.
Write as: eight thousand one hundred forty nine
kozhevka@AbobaLaptop:~/programming-kozhevnikov/lab07/build$ ./main.bin
Generated random number: 1717899131
1 < 7; 1 < 7; 7 < 8; 8 < 9; 1 < 3;
Current less next iteration count: 5
Randomly generated number 9131.
Write as: nine thousand one hundred thirty one
kozhevka@AbobaLaptop:~/programming-kozhevnikov/lab07/build$
```

1. Згідно завдання використав рандом() заміст ввода числа користувачем. В якості основи для рандому обрав час.

```
    int number = getRandom();
    printf("Generated random number: %d \n", number);
```

```
int getRandom() //Генерація сіду для  рандому
{
    srand((unsigned int)time(NULL));

    int result = (int)random();

    return result;
}
```

2. Реалізував завдання 7ї лабораторної. За число брав згенероване попереднім рандомом число

```
  int lab7result = taskFunction(number);
  printf("\nCurrent less next iteration count: %d \n", lab7result);
```

```
int taskFunction(int number) //Функція, що визначає, скільки серед заданої
{
    int arraySize = getNumberDigitsCount(number);
    int *numberArray = numberToIntArray(number,  numberMaxSize: arraySize);

    int counter = 0;
    int previousNumber = numberArray[0];

    for (int i = 1; i < arraySize; i++)
    {
        if (previousNumber < numberArray[i])
        {
            printf("%d < %d; ", previousNumber, numberArray[i]);
            counter++;
        }
        previousNumber = numberArray[i];
    }

    return counter;
}
```

3. Роботу з лабораторної №6 розбив на методи та поетапно викликав методи.
   Якщо метод повертав масив, у завершенні операції очищається пам'ять free().

```
int *numberArray = numberToIntArray(number,  numberMaxSize: maxNumbersCount); //ініціалізація масиву

int numberDecimalCount = getNumberDigitsCount(number);
int generatedPrintableNumber = combineArrayToInt( numbersArray: numberArray,  arraySize: maxNumbersCount);
char *numberAsString = numberToCharArray( numbersArray: numberArray);

printf("Randomly generated number %d. \nWrite as: %s \n",generatedPrintableNumber, numberAsString);


free(numberArray);
free(numberAsString);
```

```
int * numberToIntArray(int number, int numberMaxSize)
{
    if (numberMaxSize == 0) //get current number digits count.
    {
        numberMaxSize = getNumberDigitsCount(number);
    }

    int *resultInt = malloc((unsigned long)numberMaxSize * sizeof(int));

    for (int i = numberMaxSize - 1; i >= 0; i--)
    {
        resultInt[i] = number % 10; //Присвоюємо цифру індексу в масиві
        number /= 10;
    }

    return resultInt;
}
```

```
int getNumberDigitsCount(int number)
{
    int arraySize = 0;

    int n = number;

    do {
        n /= 10;
        arraySize++;
    } while (n != 0);

    return arraySize;
}
```

```
int combineArrayToInt(int* numbersArray, int arraySize) // Маси
{
    int result = 0;

    int numberSize = 1;

    for (int y = 1; y < arraySize; y++)
    {
        numberSize *= 10;
    }

    for (int i = 0; i < arraySize; i++, numberSize /= 10)
    {
        result += numbersArray[i] * numberSize;
    }
    return result;
}
```

```
char * numberToCharArray(int* numberArray) // Отримати число у вигляді слів.
{
    char result[100] = ""; // Рядок для зберігання словесного представлення числа


    if (numberArray[0] != 0) // Якщо у числі є тисячі
    {
        sprintf(result, "%s %s" , units[numberArray[0]], hundreds[1]);
    }

    if (numberArray[1] != 0) //Якщо у числі є сотні
    {
        sprintf(result, "%s %s %s", result, units[numberArray[1]], hundreds[0]);
    }

    if (numberArray[2] >= 2) //Якщо десятки більше 20ти включно
    {
        sprintf(result, "%s %s %s" , result , tens[numberArray[2]], units[numberArray[3]]);
    }
    else
    {
        if (numberArray[2] * 10 + numberArray[3] > 10) // Якщо десятки в межах 11-19
        {
            sprintf(result, "%s %s" , result , teens[numberArray[3]]);
        } else // немає десятків
        {
            sprintf(result, "%s %s" , result , units[numberArray[3]]);
        }
    }

    char *resultString = (char *)malloc(100);
    for (int i = 0; i < 100; ++i)
    {
        resultString[i] = result[i];
    }

    return resultString;
}
```