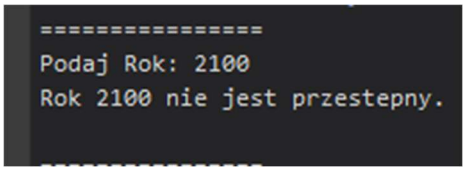
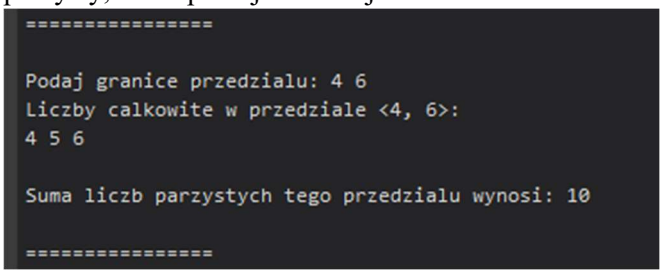
Imię i Nazwisko Patryk Twardosz	**Kierunek** Informatyka Techniczna	**Rok studiów i grupa** I rok, Gr. 5
Data zajęć Każde	**Numer i temat sprawozdania** Wszystkie	

Instrukcje warunkowe, pętle, schematy blokowe.

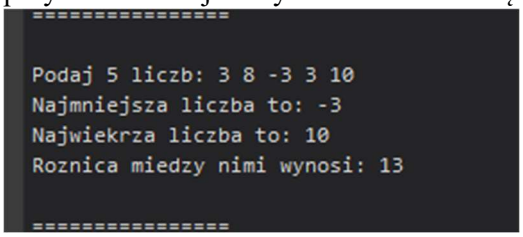
1) W celu sprawdzenia czy rok jest przestępny należy sprawdzić jego podzielność przez 4, 100 oraz 400. Jeśli jest podzielne przez 400 lub przez 4, ale nie przez 100 to jest on rokiem przestępnym.



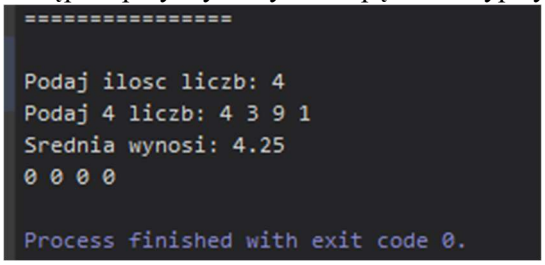
2) Program rozpoczyna wykonywać pętlę, zaczynając od mniejszej liczby, aż do dotarcia do drugiej, większej liczby. Dodatkowo poza wypisywaniem iteratora, jednocześnie jeśli jest on parzysty, do wspólnej zmiennej.



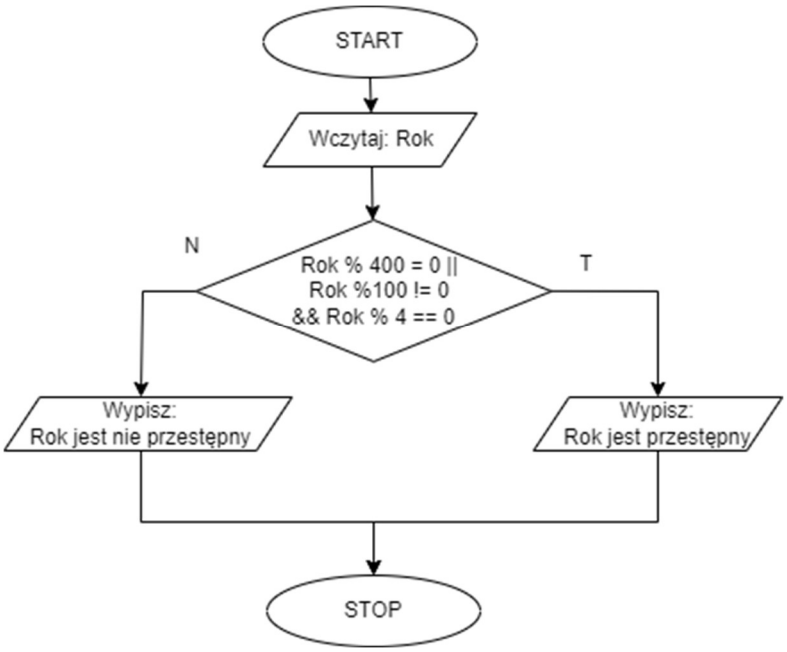
3) Zmienne przechowujące minimalną i maksymalną wartość muszą mieć odpowiednio zainicjalizowaną wartość, jak największą oraz jak najmniejszą, w celu porównywania do podawanych wartości. Podczas wczytywania danych wprowadzonych przez użytkownika, dane są od razu porównywane z obecnie największą i najmniejszą liczbą znalezioną do tej pory. Podkoniec jest wyliczana różnica między największą, a najmniejszą liczbą.



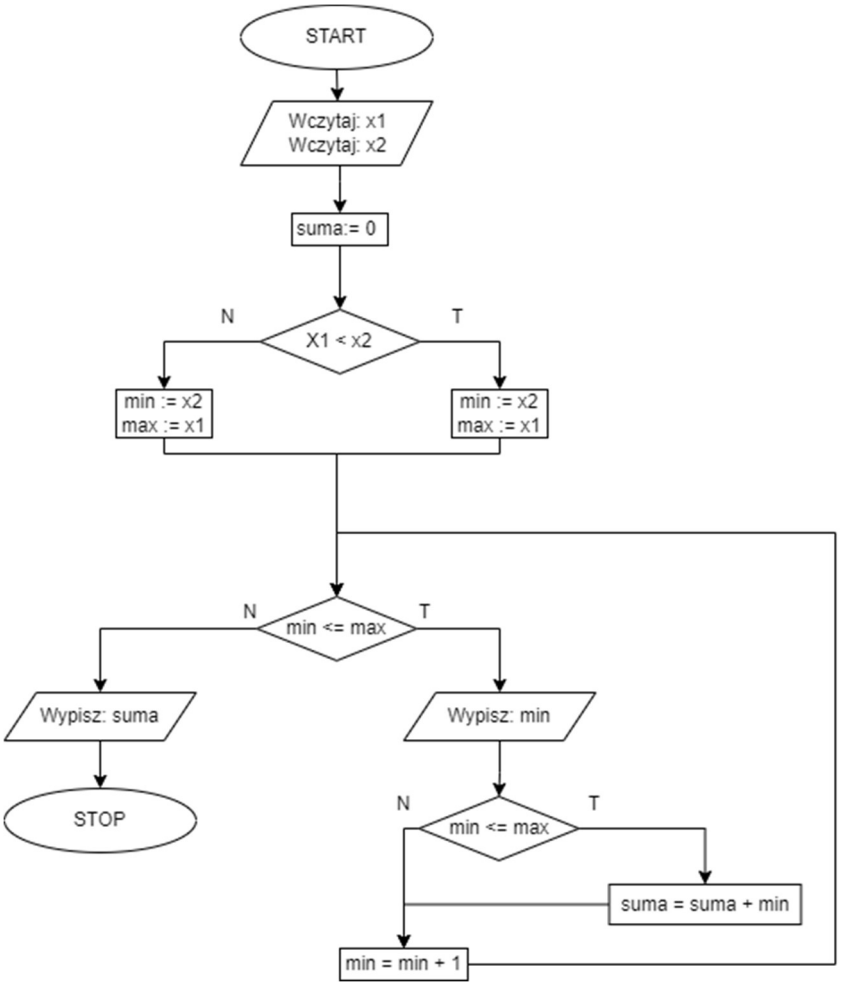
4) Najpierw program prosi o podanie ilość liczb, z których ma zostać wyliczona średnia, następnie urzytkownik pytany jest o te liczby. Program oblicza średną oraz ją wypisuje, a następnie przy wykorzystaniu pętli for wypisywana jest odpowiednia ilość ‘0’.



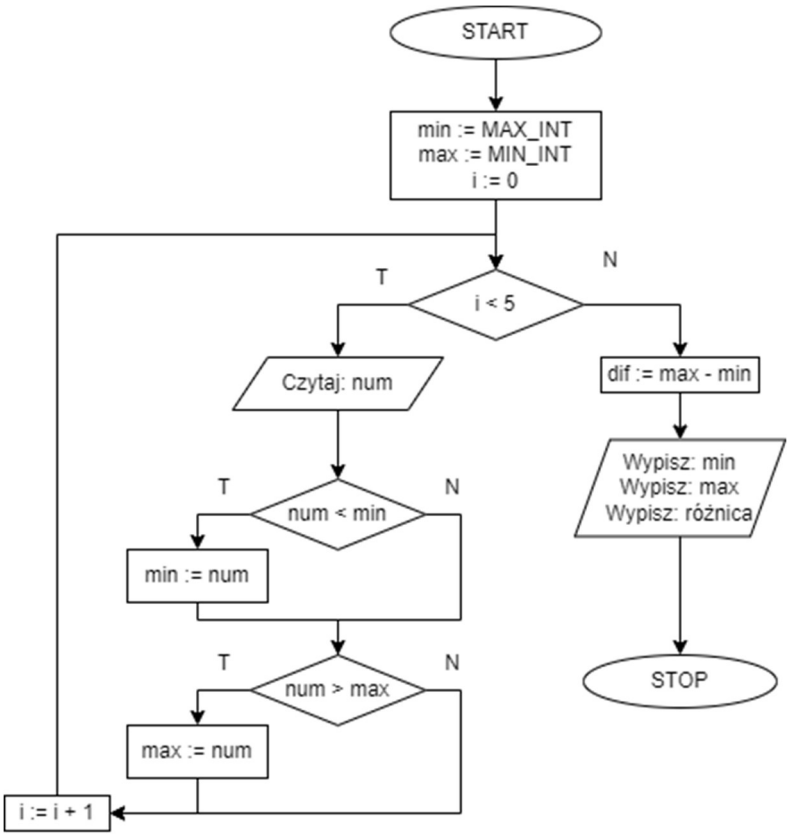
5) Schematu do powyższych zadań:
   - Zad. 1



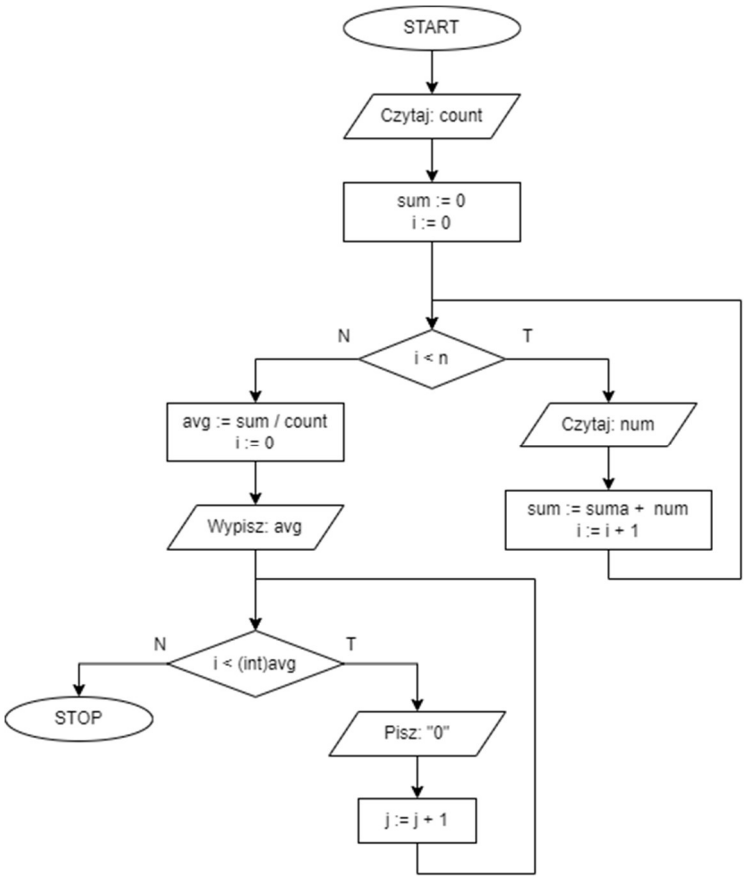
   - Zad. 2



- Zad. 3

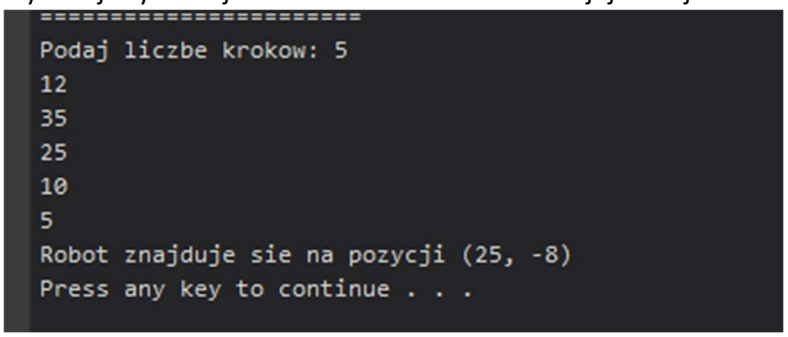

START
min := MAX_INT
max := MIN_INT
i := 0
T
i < 5
N
Czytaj: num
dif := max - min
T
num < min
N
Wypisz: min
Wypisz: max
Wypisz: różnica
min := num
T
num > max
N
STOP
max := num
i := i + 1

- Zad. 4

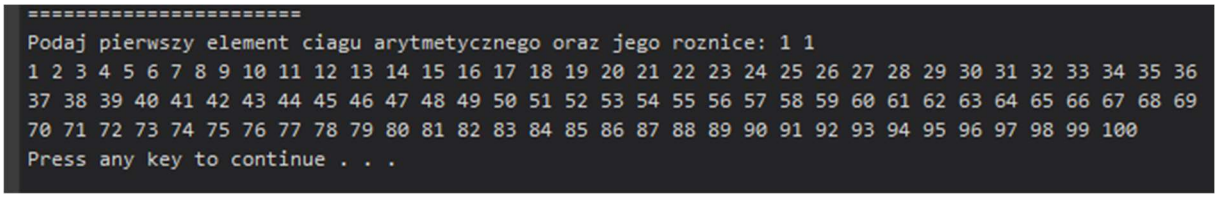

START
Czytaj: count
sum := 0
i := 0
N
i < n
T
avg := sum / count
i := 0
Czytaj: num
Wypisz: avg
sum := suma + num
i := i + 1
N
i < (int)avg
T
STOP
Pisz: "0"
j := j + 1

Funkcje:

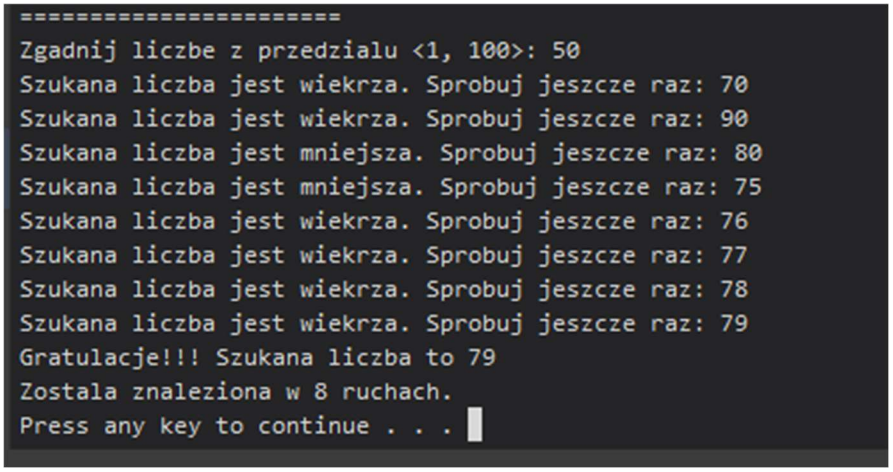
1) Użytkownik podaje liczbę kroków, które ma przejść robocik, a następnie sekwencję odległości, które przeszedł. Program przy pomocy pętli for wczytuje podane wartości oraz wykonuje symulacje kroków w zależności od swojej rotacji.



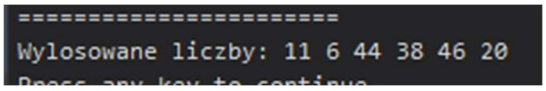
2) Po podaniu przez użytkownika pierwszego elementu oraz różnicy ciągu, program wypisze kolejne 100 elementów ciągu oraz zapisze je w tablicy.



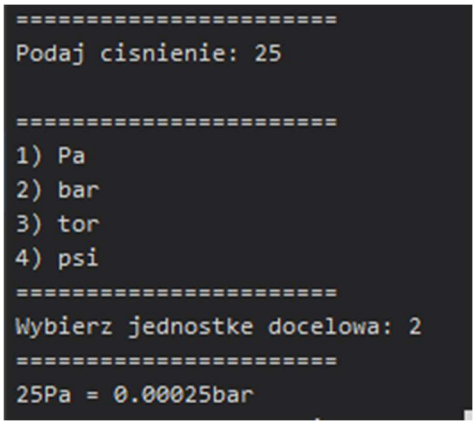
3) Po rozpoczęciu program oblicza pseudo losową liczbę, a następnie w pętli zadaje użytkownikowi pytanie, po wprowadzeniu przez użytkownika liczby, program porównuje ją do wcześniej wylosowanej liczby oraz informuje użytkownika o ich statunku. Po odgadnięciu liczby użytkownik otrzymuje gratulacje. Liczba prób zgadnięcia również jest zliczana i wyświetlana.



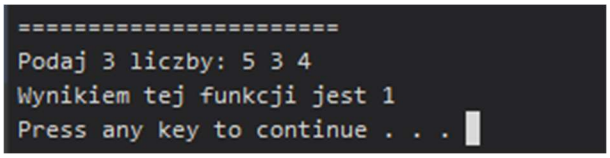
4) Program, w odtępach jedno sekundowych, wyświetli w sumie 6 unikalnych liczb z zakresu <1, 49>, symulując tym losowanie w „Totolotku”.



5) Po podaniu przez użytkownika wartości ciśnienia w Pa, program pyta się o jednostkę docelową, po czym konwertuje podaną wartość do żądanej jednostki.



6) Po podaniu danych a, b oraz c, funkcja sortuje je, po czym dokonuje sprawdzenia poprawności podanych wartości w stosunku Twierdzenia Pitagorasa.



Logika:

1.

a) $(p \vee q \Rightarrow r) \Rightarrow (p \Rightarrow r) \vee (q \Rightarrow r)$ - Jest tautologią.

p	**q**	**r**	$p \vee q$	$p \vee q \Rightarrow r$	$p \Rightarrow r$	$q \Rightarrow r$	$(p \Rightarrow r) \vee (q \Rightarrow r)$	$(p \vee q \Rightarrow r) \Rightarrow (p \Rightarrow r) \vee (q \Rightarrow r)$
0	**0**	**0**	0	0	1	1	1	1
0	**0**	**1**	0	1	1	1	1	1
0	**1**	**0**	1	0	1	0	1	1
0	**1**	**1**	1	1	1	1	1	1
1	**0**	**0**	1	0	0	1	1	1
1	**0**	**1**	1	1	1	1	1	1
1	**1**	**0**	1	0	0	0	0	1
1	**1**	**1**	1	1	1	1	1	1

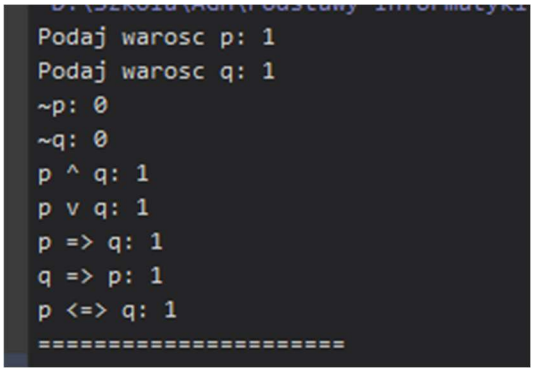
b) $(p \vee q) \wedge (p \Rightarrow q) \Rightarrow q \Rightarrow p$ - Nie jest tautologią.

p	**q**	$p \vee q$	$p \Rightarrow q$	$(p \vee q) \wedge (p \Rightarrow q)$	$(p \vee q) \wedge (p \Rightarrow q) \Rightarrow q$	$(p \vee q) \wedge (p \Rightarrow q) \Rightarrow q \Rightarrow p$
0	**0**	0	1	0	1	0
0	**1**	1	1	1	1	0
1	**0**	1	0	0	1	1
1	**1**	1	1	1	1	1

c) $\neg(p \Rightarrow q) \wedge (q \Rightarrow p) \Rightarrow p \wedge \neg q$ - Jest tautologią.

p	**q**	$\neg(p \Rightarrow q)$	$q \Rightarrow p$	$\neg(p \Rightarrow q) \wedge (q \Rightarrow p)$	$p \wedge \neg q$	$\neg(p \Rightarrow q) \wedge (q \Rightarrow p) \Rightarrow p \wedge \neg q$
0	**0**	0	1	0	0	1
0	**1**	0	0	0	0	1
1	**0**	1	1	1	1	1
1	**1**	0	1	0	0	1

2. Program pyta o podanie wartości logicznych dla p oraz q, a następnie wypisuje ich negację, koniunkcję, alternatywę, implikację oraz równoważność.



3. p – true q – true r – false

a) p ∧ q - true
b) p ∨ q - true
c) ¬p ∧ (p ∨ q) - false
d) (p ∧ r) → q - true
e) ¬(p ↔(q ∨ r)) - false
f) $[(p \rightarrow r) \vee \neg q] \leftrightarrow [p \rightarrow (r \wedge \neg q)]$ - true
g) $[(\neg r \vee q) \vee \neg(q \wedge r)] \rightarrow [\neg(q \rightarrow p)]$ - false

```
====================
Podaj wartosc p: 1
Podaj wartosc q: 1
Podaj wartosc r: 0
p ^ q: 1
p v q: 1
~p ^ (p v q): 0
(p ^ r) => q: 1
~(p <=> (q v r)): 0
[(p => r) v ~q] <=> [p => (r ^ ~q)]: 1
[(~r v q) v ~(q ^ r)] => [~(q => p)]: 0
```

Struktury danych – Grafy:

a) Odczytywanie:
- Liczba wierzchołków: 6
  Liczba krawędzi: 8
- Pary:
  - 0 -> 1
  - 1 -> 2
  - 2 -> 2
  - 1 -> 3
  - 3 -> 1
  - 2 -> 4
  - 4 -> 0
  - 4 -> 3

b) Po podaniu danych (ilości wierzchołów I krawędzi oraz połączeń między wierzchołkami) program przedstawi macierz sąsiedztwa w postaci tabeli, gdzie x to wierzchołek początkowy, a y to wierzchołek końcowy.

c) Aby program przedstawiał macierz sąsiedztwa grafu niekierunkowego poza zaznaczeniem sąsiedztwa z *x* do *y*, należy zaznaczyć sąsiedztwo z *y* do *z*.

```
Podaj liczbe wierzchoklkow:
6
Podaj liczbe krawedzi:
8
Podaj pary wierzcholkow:
0 1
1 2
2 2
1 3
3 1
2 4
4 0
4 3

   0 1 2 3 4 5
  ------------
0| 0 0 0 0 1 0
1| 1 0 0 1 0 0
2| 0 1 1 0 0 0
3| 0 1 0 0 1 0
4| 0 0 1 0 0 0
5| 0 0 0 0 0 0
```

Systemy i reprezentacja liczb:

1.

a) 113(10) = 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^0 = 1110001(2)
432(10) = 1 * 2^8 + 1 * 2^7 + 1 * 2^5 + 1 * 2^4 = 110110000(2)

b) 555(10) = 2 * 16^2 + 2 * 16^1 + 11 * 16^0 = 22B(16)
8736(10) = 2 * 16^3 + 2 * 16^2 + 2 * 16^1 = 2220(16)

c) 1_0011(2) = 13(16)
1_0100_1011(2) = 14B(16)

d) D5E7(16) = 13 * 16^3 + 5 * 16^2 + 14 * 16^1 + 7 = 54759(10)
F01A33(16) = 15 * 16^5 + 1 * 16^3 + 10 * 16^2 + 3 * 16^1 + 3 = 15735347(10)

e) 752(8) = 1_1110_1010(2) = 1EA(16)
2641(8) = 0101_1010_0001(2) = 5A1(16)

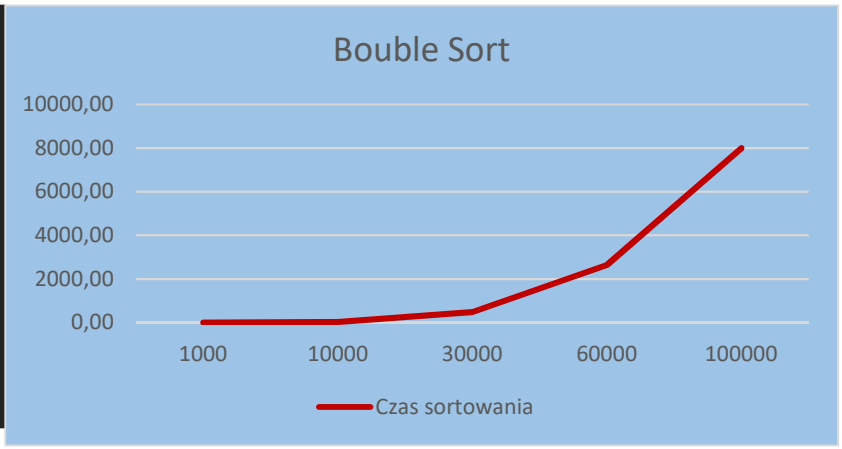
2. Program konwertuje przykładowe dane wejściowe:

```
dec2bin: 44(10) = 101100(2)
dec2hex: 44(10) = 2C(16)
bin2dec: 101(2) = 5(10)
bin2hex: 101100(2) = 2C(16)
oct2bin: 115(8) = 001001101(2)
```

Złożoność obliczeniowa:

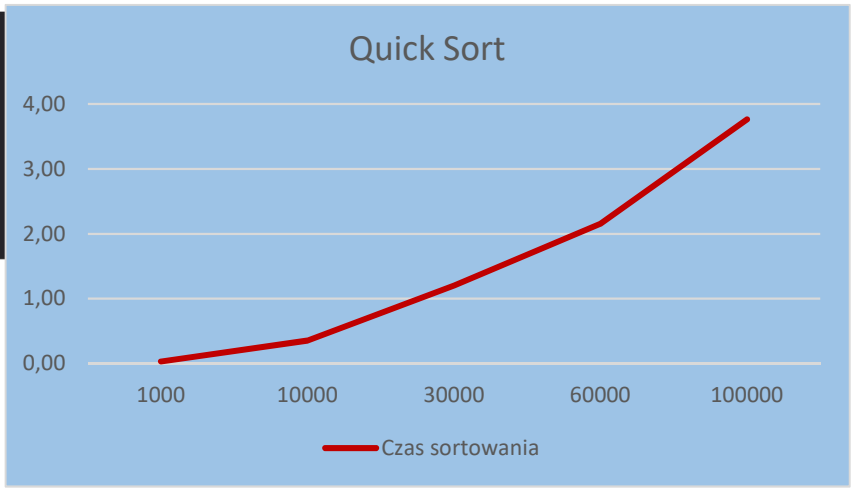
1. Bouble Sort (ms) – $O(n^2)$

```
============================
1) Bubble Sort
2) Quick Sort
3) Heap Sort
4) Selection Sort
============================
Wybierz algorytm sortowania: 1
======================

Bubble Sort
Sorting of 1000 elements sorted in: 0.364ms
Sorting of 10000 elements sorted in: 34.0669ms
Sorting of 30000 elements sorted in: 484.983ms
Sorting of 60000 elements sorted in: 2637.55ms
Sorting of 100000 elements sorted in: 7996.33ms
Press any key to continue . . .
```



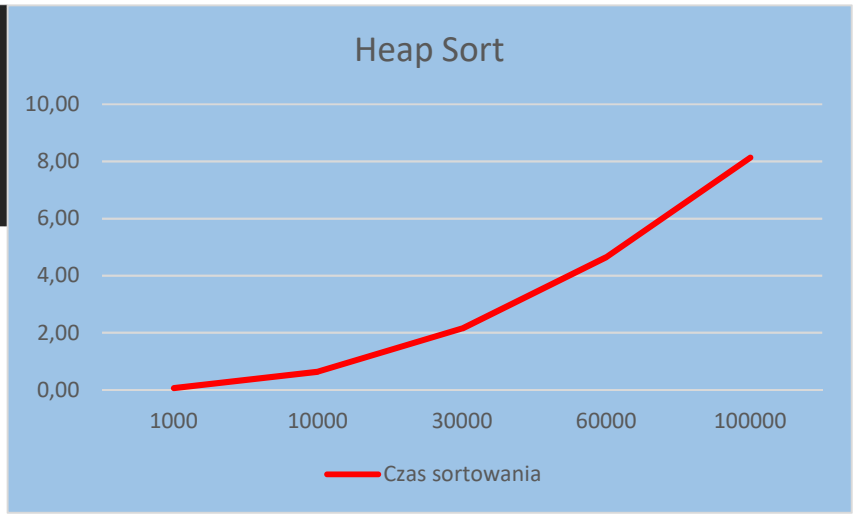
2. Quick Sort (ms) – O(n * log(n))

```
======================

Quick Sort
Sorting of 1000 elements sorted in: 0.0324ms
Sorting of 10000 elements sorted in: 0.3572ms
Sorting of 30000 elements sorted in: 1.2017ms
Sorting of 60000 elements sorted in: 2.1556ms
Sorting of 100000 elements sorted in: 3.7616ms
Press any key to continue . . .
```



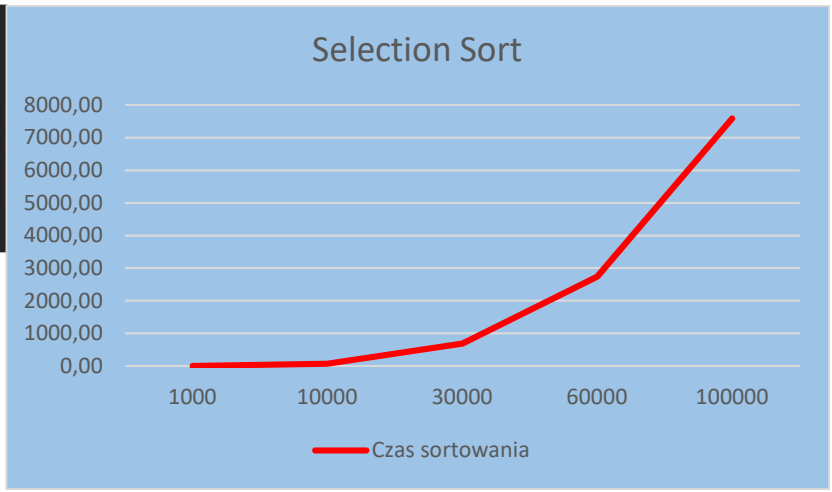
3. Heap Sort (ms) – O(n * log(n))

```
Heap Sort
Sorting of 1000 elements sorted in: 0.0614ms
Sorting of 10000 elements sorted in: 0.642ms
Sorting of 30000 elements sorted in: 2.1485ms
Sorting of 60000 elements sorted in: 4.6546ms
Sorting of 100000 elements sorted in: 8.1363ms
Press any key to continue . . .
```



4. Selection Sort (ms) – $O(n^2)$

```
======================

Selection Sort
Sorting of 1000 elements sorted in: 0.7441ms
Sorting of 10000 elements sorted in: 75.4833ms
Sorting of 30000 elements sorted in: 686.239ms
Sorting of 60000 elements sorted in: 2734.78ms
Sorting of 100000 elements sorted in: 7584.3ms
Press any key to continue . . .
```



Kryptografia:

Program po uruchomieniu wyświetla menu, w którym można wybrać 4 akcje (szyfrowanie i deszyfrowanie w Szyfrze Cezara oraz przy pomocy szyfrowania kluczem XOR.

Po wyborze akcji użytkownik jest proszony o podanie pliku, którego zawartość ma zostać przetworzona oraz o przesunięcie lub maskę, potrzebne do przeprowadzenia odpowiedniej operacji.