**SPRAWOZDANIE**

**PROJEKT Z PRZEDMIOTU**

**„ALGORYTMY I STRUKTURY DANYCH”**

**TEMAT**

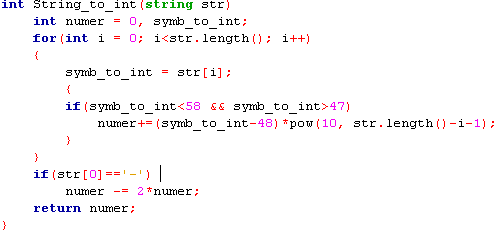
„Wypisz elementy tablicy dwuwymiarowej w rozmiarze MxN obchodząc ją po spirali”

Bartłomiej Trzepacz, PRz, semestr 2020/2021, Inżynieria i Analiza Danych

# 1.Wstęp

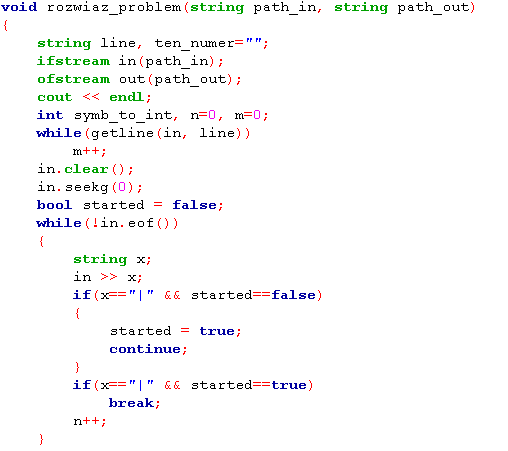
Sprawozdanie dotyczy projektu z przedmiotu Algorytmy   
i Struktury Danych. Celem projektu było zaimplementowanie algorytmu, który dla zadanej tablicy A o rozmiarze MxN różnych liczb całkowitych „rozwija ją po spirali” po czym wypisuje liczby w jednej linii w nowym pliku tekstowym.

# 2. Opis problemu



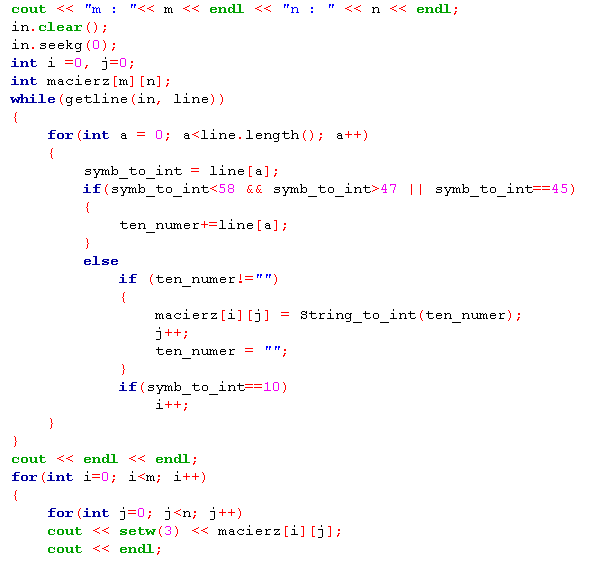
Rys. 1 Część 1 programu

Program rozpoczyna pracę od zmiany zmiennych typu string na „int” za pomocą tabeli kodu ASCII.



Rys. 2 Część 2 programu

Kolejnym krokiem jest stworzenie funkcji, która otwiera plik wejściowy i wyjściowy. Zlicza ona liczbę wierszy oraz kolumn za pomocą pętli while. Dzieje się to dzięki funkcji if która rozpoznając znak „|” w pliku wejściowym rozpoczyna wiersz oraz kończy go po znalezieniu drugiego takiego samego znaku po czym zwiększa m.



Rys. 3 Część 3 programu

Następnie wypisujemy rozmiar naszej macierzy. W dalszej części programu za pomocą pętli while poruszamy się po tablicy z pliku wejściowego i odczytujemy zapisane w niej znaki które kolejno zapisujemy w naszym algorytmie. Po zakończeniu tego kroku wypisujemy macierz w programie w takiej samej postaci w jakiej jest ona zapisana w pliku wejściowym.

# 

Rys. 4 Część 4 programu

Kolejnym elementem algorytmu jest utworzenie kierunków, po których będziemy się poruszać w celu rozwinięcia tablicy po spirali.

# 3.Pseudokod

zdefiniuj funkcja string\_to\_int (przekaż do funkcji ciąg znaków str)

{

numer <- 0

zainicjuj liczbę symb\_to\_int

i <- 0

dla i mniejszego od długości str, wykonuj i zwiększ i o 1

{

symb\_to\_int <- str[i]

jeśli symb\_to\_int jest pomiędzy większy od 47 i mniejszy od 58 to

{

zwiększ numer o (symb\_to\_int - 48) \*10^ (długości str - i -1)

}

}

jeśli pierwszy znak str to '-', to

{

pomniejsz numer o jego dwukrotność

}

zwróć numer

}

zdefiniuj funkcje rozwiaz\_problem (przekaż do funkcji ciągi znaków path\_in, path\_out)

{

zainicjuj ciągi znaków line, ten\_numer

otwórz path\_in do odczytu jako in

otwórz path\_out do zapisu jako out

przejdź do nowej linii w konsoli

zainicjuj liczbę symb\_to\_int

n <- 0

m <- 0

dopóki nie dojdzie do ostatniej linii w in

{

zwiększ m o 1

}

wróć wskaźnik w in na jego początek

started <- fałsz

dopóki nie dojdzie do końca in

{

inicjuj ciąg znaków x

wczytaj znak z in do x

jeśli x to "|" i status started to fałsz to

{

started <- prawda

kontynuuj pętle

}

jeśli x to "|" i status started to prawda to

{

wyjdź z pętli

}

zwiększ n o jeden

}

napisz "m: /wartość m/"

napisz "n: /wartość n/"

ustaw kursor w in na jego początek

i <- 0

j <- 0

utwórz tabele o nazwie macierz o rozmiarach m x n

dopóki nie dojdzie do ostatniej linii w in

{

line <- kolejna linia w in

a <- 0

dla a mniejsze niż długość line wykonuj i zwiększ a o 1

{

symb\_to\_int <- line[a]

jeśli symb\_to\_int jest większy niż 47 i mniejszy niż 58, albo równy 45

{

dodaj line[a] na koniec ten\_numer

}

w innym razie

{

jeśli ten\_numer różni się od ""

{

macierz[i][j] <- wynik funkcji string\_to\_int(ten\_numer)

}

jeśli

{

symb\_to\_int jest równy 10 to

{

zwiększ i o 1

}

}

}

}

}

przejdź 2 linie niżej w konsoli

i <- 0

dla i mniejszego od m wykonaj i zwiększ i o 1

{

j <- 0

dla j mniejszego od n wykonaj i zwiększ j o 1

{

przesuń kursor w konsoli o 3 w prawo

wypisz macierz[i][j]

}

przejdź linie niżej w konsoli

}

i <- 0

j <- 0

alfa <- 0

k <- 0

zainicjuj ciąg znaków output

wszystkie\_kierunki [][2] <- {{0,1}, {1,0}, {0,-1}, {-1,0}}

zainicjuj liczbę obecny\_kierunek

dopóki alfa jest większe od a i alfa jest mniejsza od n

{

zwiększ alfa o resztę z dzielenia k przez 2

obecny\_kierunek <- reszta z dzielenie k przez 4

tabela długość [2] <- {0,0}

dopóki długość [0] jest mniejsza od różnicy m i alfa oraz długość [1] jest mniejsza od różnicy n i alfa to

{

zwiększ i o wszystkie\_kierunki [obecny\_kierunek [0]]

zwiększ j o wszystkie\_kierunki [obecny\_kierunek [1]]

zwiększ długość [0] o wartość bezwzględną wszystkie\_kierunki [obecny\_kierunek [0]]

zwiększ długość [1] o wartość bezwzględną wszystkie\_kierunki [obecny\_kierunek [1]]

przesuń kursor w konsoli o 3 w prawo

wypisz macierz[i][j]

dodaj na koniec output to\_string(macierz[i][j]) + " "

}

zwiększ k o 1

}

jeśli m jest większe od n oraz n i alfa są sobie równe

{

tabela długość [2] = {0,0}

dopóki długość [0] jest mniejsza od różnicy m i alfa to

{

zwiększ i o wszystkie\_kierunki [obecny\_kierunek [0]]

zwiększ j o wszystkie\_kierunki [obecny\_kierunek [1]]

zwiększ długość [0] o wartość bezwzględną wszystkie\_kierunki [obecny\_kierunek [0]]

zwiększ długość [1] o wartość bezwzględną wszystkie\_kierunki [obecny\_kierunek [1]]

przesuń kursor w konsoli o 3 w prawo

wypisz macierz[i][j]

dodaj na koniec output to\_string(macierz[i][j]) + " "

}

}

zapisz do out

}

główna część programu

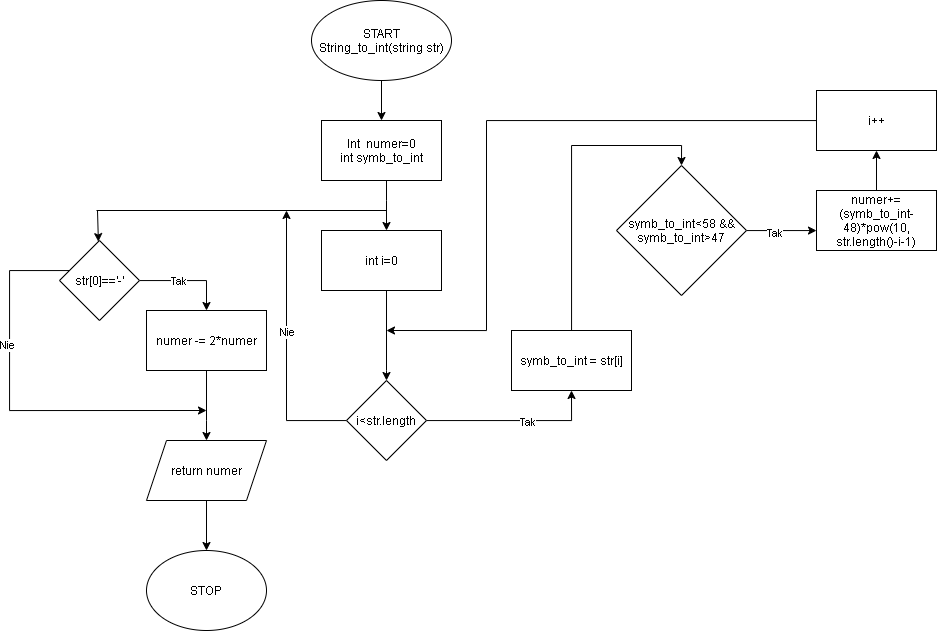
{

wywołaj funkcję rozwiaz\_problem ("input.txt", "output.txt")

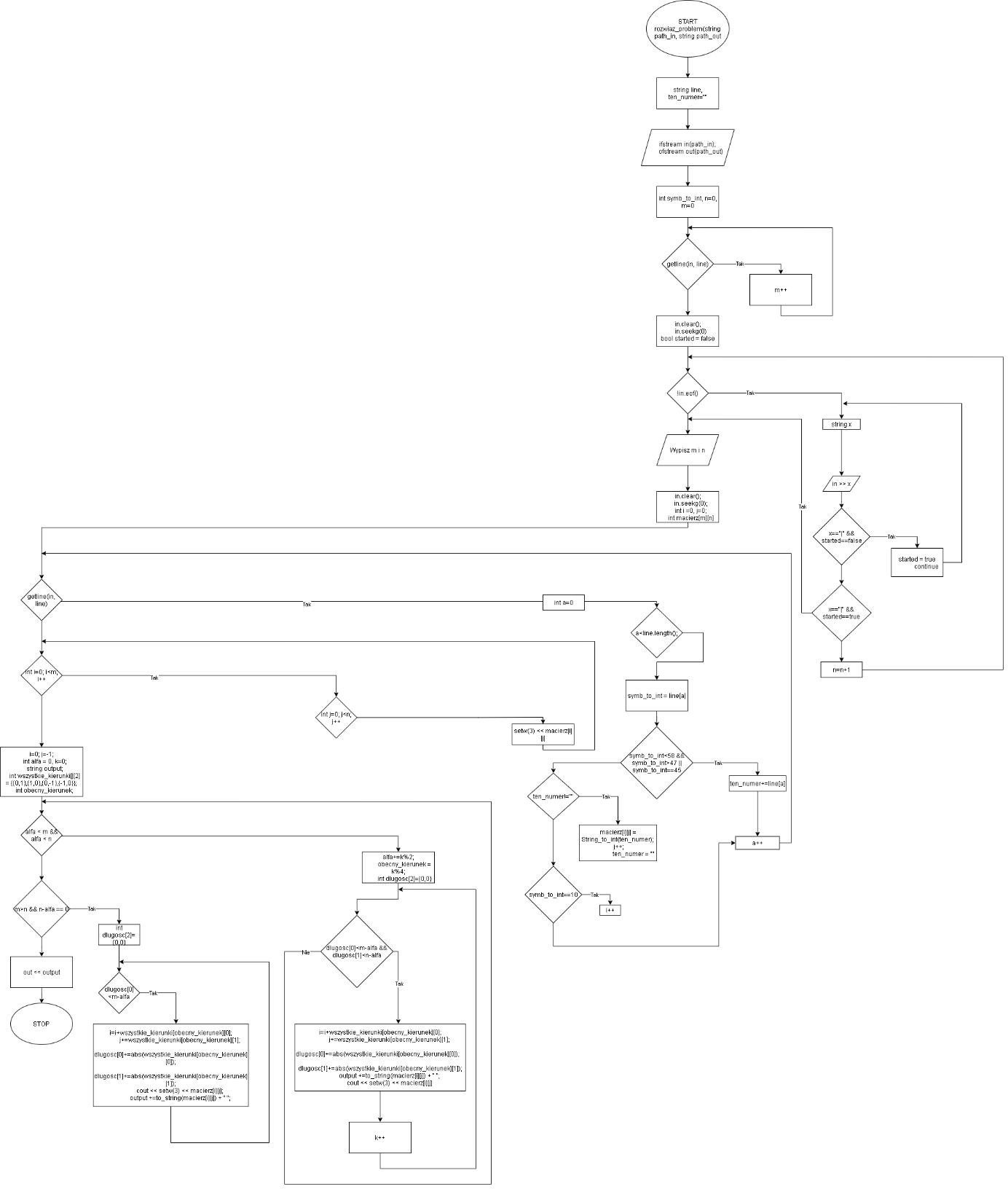
zakończ program

}

# 4. Schemat Blokowy



Rys. 5 Schemat blokowy pierwszej funkcji



Rys. 6 Schemat blokowy drugiej funkcji

# 

Rys. 7 Schemat blokowy trzeciej funkcji

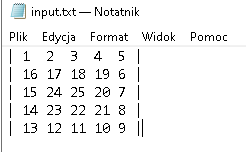
# 

# 5. Dokumentacja z doświadczeń

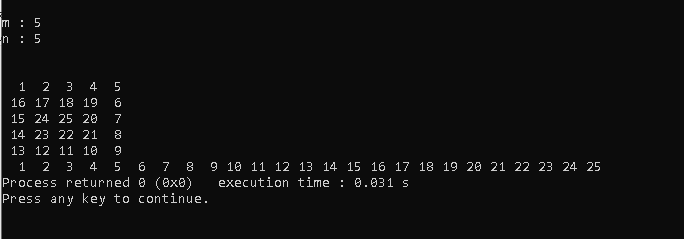
Przeprowadziłem trzy testy dla kolejno: liczb dodatnich, ujemnych oraz mieszanych.

Ad. Test 1

Program pobrał dane na temat dwuwymiarowej tablicy A z liczbami dodatnimi o wymiarach 5x5 z pliku tekstowego o nazwie input.txt, algorytm stworzył tablice B która jest rozwinięciem tablicy A obchodząc ją po spirali po czym wypisał ją w programie oraz zapisał w nowym pliku tekstowym o nazwie output.txt



Rys. 8 input.txt (1)



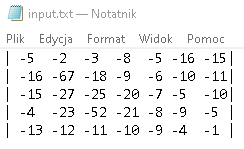
Rys. 9 Test na tablicy o wymiarach 5x5 z liczbami dodatnimi



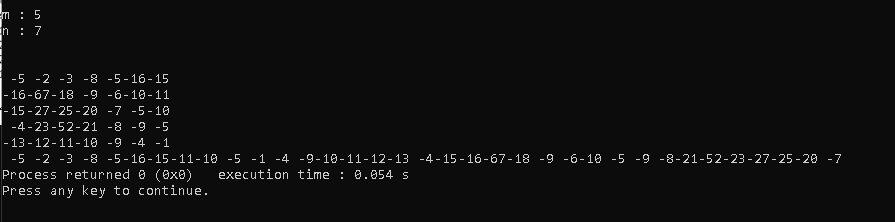
Rys.10 output.txt (1)

Ad. Test 2

Program pobrał dane na temat dwuwymiarowej tablicy A z liczbami ujemnymi o wymiarach 5x7 z pliku tekstowego o nazwie input.txt, algorytm stworzył tablice B która jest rozwinięciem tablicy A obchodząc ją po spirali po czym ponownie wypisał ją w programie oraz zapisał w nowym pliku tekstowym o nazwie output.txt



Rys.11 input.txt (2)



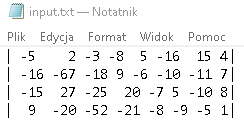
Rys. 12 Test na tablicy o wymiarach 5x7 z liczbami ujemnymi



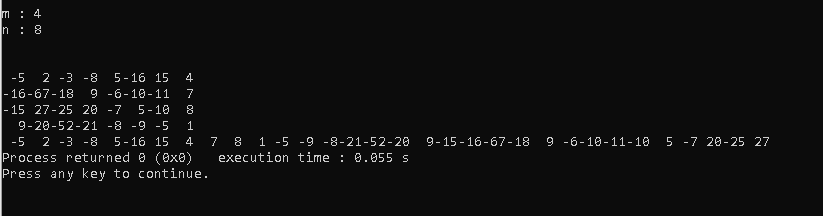
Rys.13 output.txt (2)

Ad. Test 3

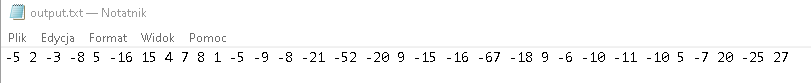
Program pobrał dane na temat dwuwymiarowej tablicy A z liczbami ujemnymi oraz dodatnimi o wymiarach 4x8 z pliku tekstowego o nazwie input.txt, algorytm stworzył tablice B która jest rozwinięciem tablicy A obchodząc ją po spirali po czym ponownie wypisał ją w programie oraz zapisał w nowym pliku tekstowym o nazwie output.txt



Rys.14 input.txt (3)



Rys.15 Test na tablicy o wymiarach 4x8 z liczbami ujemnymi oraz dodatnimi



Rys. 16 output.txt (3)

# 6.Wnioski

Projekt został zrealizowany, działanie algorytmu jest poprawne dla liczb dodatnich jak i ujemnych. W programie główny algorytm został zaimplementowany w osobnej funkcji, która jest wywoływana w późniejszych etapach działania programu. Projekt posiada możliwość odczytu z pliku tekstowego jak i późniejsze zapisanie wyników do pliku tekstowego. Testy uwidocznione za pomocą zrzutów ekranu w 5. Dokumentacja z doświadczeń udowadniają poprawne działanie programu. W programie zostały umieszczone stosowane komentarze, które pomagają w zrozumieniu kodu również w sprawozdaniu zostały umieszczone elementy które ułatwiają nam zrozumienie działania algorytmu szukającego jest to pseudokod i schemat blokowy. Najcięższe w tym zadaniu było napisanie funkcji, która umożliwia odczytywanie danych z pliku, ale po dużej ilości prób udało się zrealizować to zagadnienie.