



**POLITECHNIKA
RZESZOWSKA**
im. IGNACEGO ŁUKASIEWICZA

Karolina Magdoń

Inżynieria i analiza danych
I rok

Sprawozdanie z projektu przesuwania elementów tablicy (C++)

Praca przygotowana na zajęcia
„Algorytmy i struktury danych”
w roku akademickim 2022/2023

Rzeszów, listopad 2022

Spis treści

1. Wstęp i opis zagadnień projektu	3
2. Podstawy teoretyczne zagadnienia	3
3. Cechy programu	4
4.1 Złożoność obliczeniowa i czasowa	5
4.2 Schemat blokowy	6
4.3 Dane we/wy i zmienne pomocnicze	7
4.4 Rezultaty testów	8
5. Podsumowanie i wnioski	10
6. Appendix: kod programu	11

Spis ilustracji

Rys. 1 Schemat blokowy algorytmu	6
Rys. 2 Test 1. (wcześniejsza wersja programu)	8
Rys. 3 Test dla tablicy pobieranej z pliku tekstowego (dane z przykładu dołączonego do zadania)	8
Rys. 4 Test 2. dla tablicy długości 10 liczb z przedziału [-100,100]	8
Rys. 5 Test 3. dla tablicy długości 100 liczb z przedziału [-200,200]	8
Rys. 6 Test 4. dla tablicy długości 1000 liczb z przedziału [-5000,5000]	9
Rys. 7 Wykres długości czasu działania przy zwiększającej się długości tablicy	9

1. Wstęp i opis zagadnień projektu

Moim zadaniem projektowym było utworzenie programu, który dla zadanej tablicy liczb całkowitych przesunie wszystkie elementy mniejsze od 0 na jej koniec. Dodatkowym warunkiem było zachowanie kolejności występowania elementów ujemnych. Do kolejnych zaleceń należało: przedstawienie podstaw teoretycznych zagadnienia oraz schemat blokowy algorytmu (również pseudokod). Zobrazować rezultaty działania programu. Ukazać złożoność czasową, obliczeniową.

2. Podstawy teoretyczne zagadnienia

W algorytmie będę używać Tablic i wskaźników, ważne jest, by rozumieć, co znaczą te pojęcia.

Tablica - Tablica to zbiór danych określonego typu. Dostęp do danych możliwy jest za pomocą kluczy, będących ich indeksami. Wykorzystanie tablic umożliwia operowanie na dużych ilościach danych tego samego typu. Deklaracji dokonujemy w sposób następujący:

typ_danych nazwa_tablicy[ilość_elementów];

Wskaźnik - Wskaźnik jest zmienną przechowującą adres innej zmiennej. Wskaźnik przechowuje adres zmiennej znajdującej się w pamięci komputera. Nie ma konieczności odwoływania się do konkretnej zmiennej. Deklaracja wskaźnika wygląda prawie identycznie jak deklaracja typowej zmiennej:

*int *wskaznik;*

Dla nas wskaźniki będą o tyle przydatne, że w przeciwieństwie do Tabeli, gdzie zawsze trzeba podać indeks, do którego się odwołujemy, wskaźnik będzie można w całości przekazać jako to co zwraca funkcja. Bez tego nie mogłabym działać większą ilością funkcji na jednej tabeli.

3. Cechy programu

1. Możliwość odczytywania danych wejściowych z pliku tekstowego i zapisu posortowanego ciągu do pliku tekstowego z danymi wyjściowymi.
2. Na potrzeby testów zaimplementowanie funkcji generującej „losowe” ciągi elementów (o zadanej długości)
3. Główna funkcja przesuująca elementy w tablicy tak, by spełniały wymagania zadania.
4. Mierzenie czasu działania głównego algorytmu i zapis do pliku czasu działania dla serii zmierzonych czasów z zwiększającą się ilością elementów tablic.

Do wykonania tego projektu stworzyłam kod w języku C++ zawierający:

- funkcję wczytującą z pliku serię liczb,
- funkcję generującą losowy zbiór liczb (różną ich ilość oraz liczby z różnych przedziałów) służącą do testowania algorytmu przesuującego,
- właściwy algorytm przesuujący,
- zaimplementowany do algorytmu przesuującego miernik czasu działania,
- segmenty zapisujące do plików otrzymane dane.

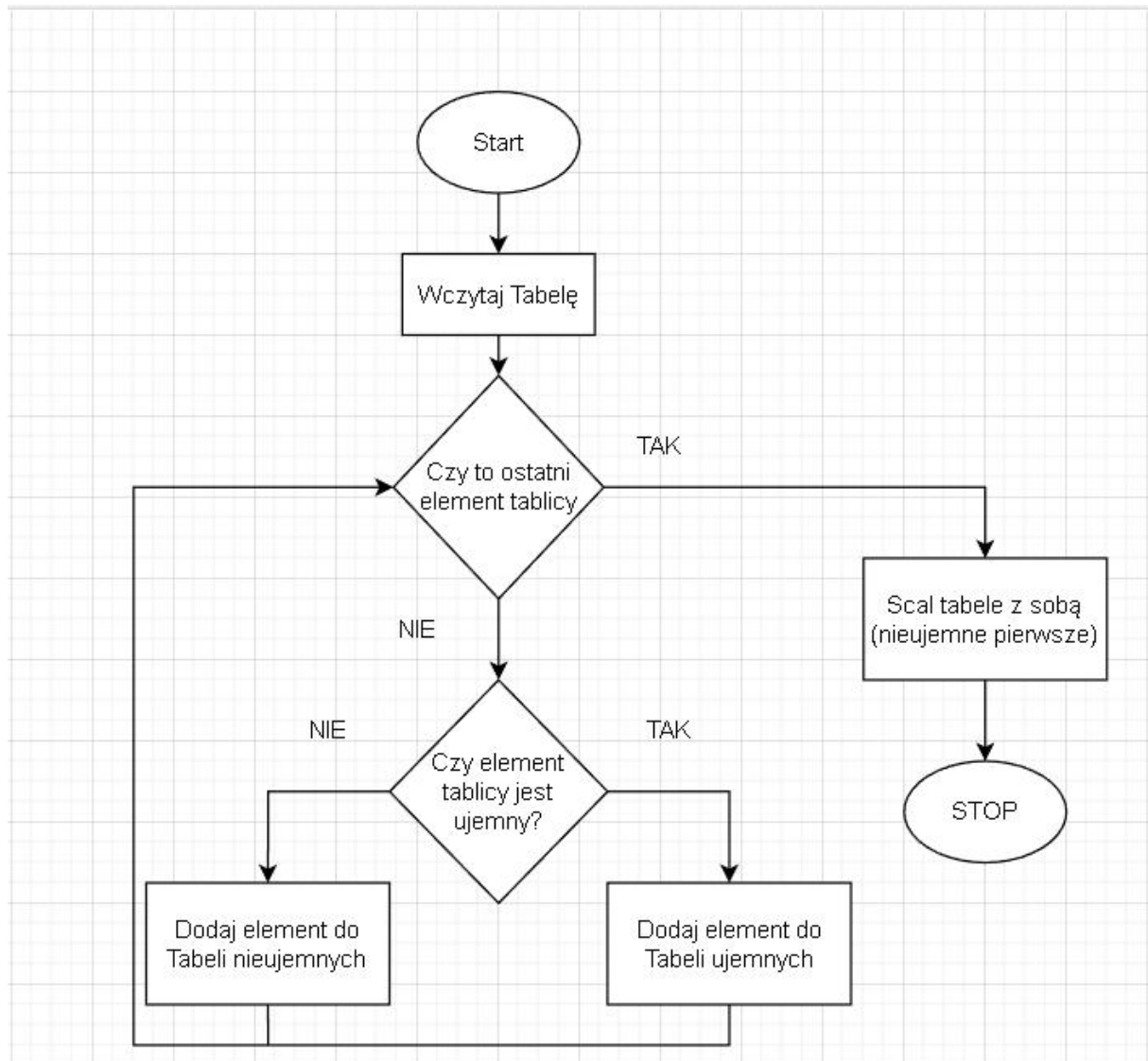
4. Opis algorytmu przesuwającego

Algorytm bierze Tablicę liczb naturalnych i dla każdego elementu sprawdza, czy jest ujemna czy nieujemna. Wstawia go do odpowiedniej tablicy tymczasowej, po czym scala te tablice, zaczynając od tablicy elementów nieujemnych.

4.1 Złożoność obliczeniowa i czasowa

Złożoność obliczeniowa algorytmu jest bardzo wydajna, w każdym wypadku wynosi $O(n)$, gdyż wystarczy, że tylko 2 razy przeanalizujemy wszystkie liczby z tabeli, aby wyznaczyć wynik. Złożoność czasowa jest znikoma, miałam problemy ze sprawieniem, by miernik czasu nawet zmierzył czas wykonywania się algorytmu. W najgorszym wypadku będzie to kilka tysięcznych sekundy, przy naszej skali działania jest to prawie nic.

4.2 Schemat blokowy



Rys. 1 Schemat blokowy algorytmu

4.3 Pseudokod

K01: Wczytaj Tabelę

K02: Stwórz 2 Tabele Pomocnicze:

1. Tabela która będzie przetrzymywać liczby nieujemne
2. Tabela która będzie przetrzymywać liczby ujemne

K03: Stwórz 2 Zmienne Pomocnicze:

1. Zmienna która będzie wskaźnikiem tabeli z liczbami nieujemnymi
2. Zmienna która będzie wskaźnikiem tabeli z liczbami ujemnymi

K04: Patrzymy na każdy element po kolei i sprawdzamy dla niego, czy jest mniejszy od 0, jeśli nie, przechodzimy do K04.1, jeśli tak, do K04.2

K04.1 Dopisz element do Tabeli dla elementów nieujemnych na miejsce, które wskazuje wskaźnik i zwiększ wartość wskaźnika tabeli z liczbami nieujemnymi

K04.2 Dopisz element do Tabeli dla elementów ujemnych na miejsce, które wskazuje wskaźnik i zwiększ wartość wskaźnika tabeli z liczbami ujemnymi

K05: Scalamy Tabele ze sobą (nieujemne jako pierwsza)

K06: Wypisz Tabelę.

4.3 Dane we/wy i zmienne pomocnicze

- Dane wejściowe: Tablica nieposortowana;
- Dane wyjściowe: Tablica posortowana;
- Zmienne pomocnicze:
 - Tablice Ujemnych/Nieujemnych – służą do przechowywania rozdzielonych liczb;
 - Iteratory Ujemnych/Nieujemnych - do zapisywania do tablic i do liczenia ich długości;
 - Przesunięta tablica – tablica, do której scalamy tablice nieujemnych i ujemnych;
 - Iterator przesuniętej tablicy - do zapisywania do tablicy i do liczenia jej długości.

4.4 Rezultaty testów

Algorytm w każdym przypadku poprawnie wykonał przesuwanie na Tabeli. Oto parę przykładów:

```
Wygenerowana Tablica = [ 6 -7 8 4 -1 -7 9 8 -3 9 ]
Tablica po przesunięciu = [ 6 8 4 9 8 9 -7 -1 -7 -3 ]
```

Rys. 2 Test 1. (wcześniejsza wersja programu)

```
Wczytana Tablica = [ -10 5 8 -4 1 3 0 -7 ]
Tablica po przesunięciu = [ 5 8 1 3 0 -10 -4 -7 ]
```

Rys. 3 Test dla tablicy pobieranej z pliku tekstowego (dane z przykładu dołączonego do zadania)

```
Dlugosc: 10 Granica: 100
Wygenerowana Tablica = [ -48 14 -40 62 40 98 88 -86 -52 -82 ]
Posortowana Tablica = [ 14 62 40 98 88 -48 -40 -86 -52 -82 ]
```

Rys. 4 Test 2. dla tablicy długości 10 liczb z przedziału [-100,100]

```
Dlugosc: 100 Granica: 200
Wygenerowana Tablica = [ 52 114 -140 162 140 -2 -12 -186 48 -182 -164 104 24 120 116 112 -194 -176 178 -74 2 -80 -54 120 -174
-44 -4 102 -2 -168 -86 -24 16 10 -142 24 154 122 -60 -10 92 100 -72 182 -150 -40 -22 128 154 -168 68 -142 26 194 -130 108 -1
94 20 110 68 62 54 22 70 64 68 186 -80 0 2 160 -124 -98 50 -64 -132 -56 72 -16 -62 -110 -56 8 -70 -64 -154 198 -152 -4 -28 -9
2 180 -194 28 -70 -82 196 14 -84 -118 ]
Posortowana Tablica = [ 52 114 162 140 48 104 24 120 116 112 178 2 120 102 16 10 24 154 122 92 100 182 128 154 68 26 194 108
20 110 68 62 54 22 70 64 68 186 0 2 160 50 72 8 198 180 28 196 14 -140 -2 -12 -186 -182 -164 -194 -176 -74 -80 -54 -174 -44 -
4 -2 -168 -86 -24 -142 -60 -10 -72 -150 -40 -22 -168 -142 -130 -194 -80 -124 -98 -64 -132 -56 -16 -62 -110 -56 -70 -64 -154 -
152 -4 -28 -92 -194 -70 -82 -84 -118 ]
```

Rys. 5 Test 3. dla tablicy długości 100 liczb z przedziału [-200,200]

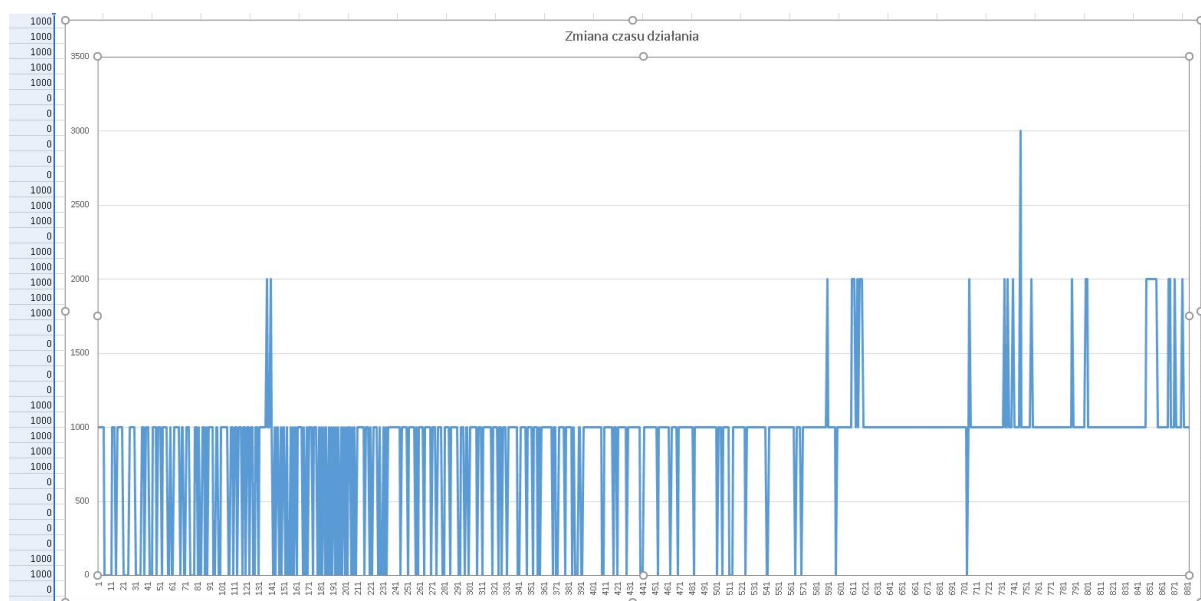

```

Dlugosc: 1000 Granica: 5000
Wygenerowana Tablica = [-2748 4514 3460 4962 -1060 -2802 -3212 3814 48 -4982 3036 4504 3224 -1080 2116 -1888 3406 1424 -2622
-2874 -3998 320 -4854 -2680 226 -4444 -4804 -3498 -802 -168 -86 -4424 -3184 4410 -2142 -1976 4554 -278 3940 -1210 -1908 -110
0 -1272 -2218 -4550 2760 -3222 928 4954 -1368 3268 -1342 -4774 -3806 1870 3308 1406 420 -3090 2468 -3538 -746 3622 -2730 464
-3932 -614 3520 -2000 1202 4560 3076 -4498 -2750 1136 2668 -456 -3528 -4416 -862 1490 -2456 808 -1670 2736 -2154 3798 -552 39
6 4772 -892 -1820 -3394 -772 -4070 4718 2196 -3986 -4084 -1318 -500 1814 4678 -3792 -4406 2054 -376 -1288 3232 -1208 1924 -44
42 -2440 3058 -1700 3324 150 -4258 386 -2858 2746 1730 -2434 1442 2594 -598 884 -160 -3970 -4628 2792 -2744 -2400 -3224 2714
1464 2768 -3502 3164 -3558 -828 4744 3896 2282 -2280 -872 3668 -518 -1526 4314 2400 -1844 -3830 -994 -2144 -2354 4632 2320 63
0 -1702 -1860 1996 3902 -3894 -4232 -1340 712 -1988 -1956 -1646 2670 582 -2734 4394 -164 686 -234 1974 -3236 -846 -1890 -180
4242 -2012 -4998 -24 696 3162 -2144 -1042 4160 2718 1326 -594 -468 3070 -4200 -1524 -3502 3230 -926 2138 -3474 -4250 -3190 -4
838 2784 -3036 1506 2416 3844 -4330 -3298 2480 -2840 -3036 4666 4752 3204 -2522 2332 -1274 -366 -2678 -4988 -3214 -410 3574 2
272 1202 -1864 382 -942 2486 -3876 -4768 2028 -2630 1556 -4484 2904 -1998 784 -3450 -3170 -3510 498 -856 -2790 3574 -4490 -34
08 -166 -1612 986 -1168 4176 4522 3768 -448 -4244 -828 3930 -4598 3352 778 326 3280 -70 -3620 -330 -2300 -4980 880 -146 3680
-4004 3400 -4016 18 -4228 -2118 -4786 -1448 -1860 -246 -3142 1104 -4048 -2414 2452 526 -1456 -1312 1350 -4348 3944 -1484 -453
0 1734 -3086 -1700 -3574 -2450 -4466 -4536 -3668 3140 -1046 -1272 -1024 2208 -4354 -754 -2612 -1006 -3266 -3542 -3868 -584 30
30 -2308 1352 -3010 -1076 -1046 4720 2236 -1632 2866 -4634 4484 -2300 -180 284 3912 -2326 762 4044 1988 4310 -4642 612 -238 -
2318 -4034 -4354 4522 -3356 -3904 -2076 -3816 -4210 -2248 -4200 3634 -2036 2064 1988 0 786 2200 -2102 -3606 1938 -546 1020 -4
012 -2092 -3020 -2322 -672 1136 136 -3266 -3988 -652 -4596 8 4864 606 3632 -2870 -3676 4636 -4728 -3552 -4950 596 -384 -3116
-4784 1980 -4988 1818 982 2058 -320 -2418 -3280 -2822 -6 466 -2616 -3174 -1658 3416 -3048 4744 -1734 4566 726 744 -938 2282 -
3658 2128 4488 -894 -1052 1138 -4962 3376 -3170 4894 998 -1110 2746 -4580 1408 224 -1666 4126 -3126 -1354 3816 -3982 -1848 -1
944 1084 -946 4308 -182 -4070 4118 -1322 -1248 -4748 2982 -228 446 -448 -3304 1256 -4330 4138 2490 1054 4950 3764 2170 2914 -
2704 -1058 2940 356 1468 -586 4888 638 -3860 4036 -382 3422 -732 816 -3898 -1730 -4448 3656 -3598 1152 143 3138 372 4094 -285
4 1250 -116 -3796 2242 -1834 2914 -3564 -4928 -674 -22 3306 -954 -2994 1
Posortowana Tablica = [-2748 4514 3460 4962 3814 48 3036 4504 3224 2116 3406 1424 320 226 4410 4554 3940 2760 928 4954 3268 1870 3
308 1406 420 2468 3622 464 3520 1202 4560 3076 1136 2668 1490 808 2736 3798 396 4772 4718 2196 1814 4678 2054 3232 1924 3058
3324 150 386 2746 1730 1442 2594 884 2792 2714 1464 2768 3164 4744 3896 2282 3668 4314 2400 4632 2320 630 1996 3902 712 2670
582 4394 686 1974 4242 696 3162 4160 2718 1326 3070 3230 2138 2784 1506 2416 3844 2480 4666 4752 3204 2332 3574 2272 1202 382
2486 2028 1556 2904 784 498 3574 986 4176 4522 3768 3930 3352 778 326 3280 880 3680 3400 18 1104 2452 526 1350 3944 1734 314
0 2208 3030 1352 4720 2236 2866 4484 284 3912 762 4044 1988 4310 612 4522 3634 2064 1988 0 786 2200 1938 1020 1136 136 8 4864
606 3632 4636 596 1980 1818 982 2058 466 3416 4744 4566 726 744 2282 2128 4488 1138 3376 4894 998 2746 1408 224 4126 3816 10
84 4308 4118 2982 446 1256 4138 2490 1054 4950 3764 2170 2914 2940 356 1468 4888 638 4036 3422 816 3656 1152 148 3138 372 409
4 1250 2242 2914 3306 -2748 -1060 -2802 -3212 -4982 -1080 -1888 -2622 -2874 -3998 -4854 -2680 -4444 -4804 -3498 -802 -168 -86
-4424 -3184 -2142 -1976 -278 -1210 -1908 -1100 -1272 -2218 -4550 -3222 -1368 -1342 -4774 -3806 -3090 -3808 -746 -2730 -3932
-614 -2000 -4498 -2750 -456 -3528 -4416 -862 -2456 -1670 -2154 -552 -892 -1820 -3394 -772 -4070 -3986 -4084 -1318 -500 -3792
-4406 -376 -1288 -1208 -4442 -2440 -1700 -4258 -2858 -2434 -598 -160 -3970 -4628 -2744 -2400 -3224 -3502 -3558 -828 -2280 -87
2 -518 -1526 -1844 -3830 -994 -2144 -2354 -1702 -1860 -3894 -4232 -1340 -1988 -1956 -1646 -2734 -164 -234 -3236 -846 -1890 -1
80 -2012 -4998 -24 -2144 -1042 -594 -468 -4200 -1524 -3502 -926 -3474 -4250 -3190 -4838 -3036 -4330 -3298 -2840 -3036 -2522 -
1274 -366 -2678 -4988 -3214 -410 -1864 -942 -3876 -4768 -2630 -4484 -1998 -3450 -3170 -3510 -856 -2790 -4490 -3408 -166 -1612
-1168 -448 -4244 -828 -4598 -70 -3620 -330 -2300 -4980 -146 -4004 -4016 -4228 -2118 -4786 -1448 -1860 -246 -3142 -4048 -2414
-1456 -1312 -4348 -1484 -4530 -3086 -1700 -3574 -2450 -4466 -4536 -3668 -1046 -1272 -1024 -4354 -754 -2612 -1006 -3266 -3542
-3868 -584 -2308 -3010 -1076 -1046 -1632 -4634 -2300 -180 -2326 -4642 -238 -2318 -4034 -4354 -3356 -3904 -2076 -3816 -4210 -
2248 -4200 -2036 -2102 -3606 -546 -4012 -2092 -3020 -2322 -672 -3266 -3988 -652 -4596 -2870 -3676 -4728 -3552 -4950 -384 -311
6 -4784 -4988 -320 -2418 -3280 -2822 -6 -2616 -3174 -1658 -3048 -1734 -938 -3658 -894 -1052 -4962 -3170 -1110 -4580 -1666 -31
26 -1354 -3982 -1848 -1944 -946 -182 -4070 -1322 -1248 -4748 -228 -448 -3304 -4330 -2704 -1058 -586 -3860 -382 -732 -3898 -17
30 -4448 -3598 -2854 -116 -3796 -1834 -3564 -4928 -674 -22 -954 -2994 1

```

Rys. 6 Test 4. dla tablicy długości 1000 liczb z przedziału [-5000,5000]

Dalej wykonałam testy sprawności zależne od ilości elementów. Ich liczba zwiększała się od 100000 do 250000, krok wynosił 100. Co dopiero, takie liczby sprawiły, że biblioteki do mierzenia czasu wykryły jakąkolwiek zmianę czasu. Wyniki wypisane zostały w milisekundach.



Rys. 7 Wykres długości czasu działania przy zwiększającej się długości tablicy

Jak widać wykres nie jest liniowy, ponieważ zależnie od losowo wygenerowanych liczb, czasami czas był mierzony, jako tysięczne sekundy, a czasami po prostu 0. Widać jednak lekką tendencję wzrostu czasu działania funkcji, być może przy ogromnych ilościach elementów czas działania byłby znaczny. (Mi niestety przy większych liczbach wywała CodeBlocks.)

5. Podsumowanie i wnioski

Założenia projektu udało się zrealizować relatywnie nieskomplikowanym obliczeniowo algorytmem, ponadto poradził on sobie bardzo dobrze, wykonanie algorytmu wręcz nie zajmuje żadnego czasu i co dopiero przy ogromnych długościach tablic mogliśmy wykryć czas działania, który i tak był liczony w tysięcznych sekundach. Przez specyfikę zagadnienia użycie algorytmu sortującego nie dawało nam pożądanego wyniku, więc użyłam w moim algorytmie bardziej codziennego znaczenia słowa sortowanie i podzieliłam liczby na ujemne i nieujemne. Ponieważ przechodziliśmy po tabelach w kolejności od lewej do prawej, tabele po „sortowaniu” mogą zostać ze sobą złożone bez dodatkowych operacji.

Podsumowując:

1. Program odczytuje dane liczbowe z pliku i zapisuje do tablicy.
2. Program zapisuje wynik do pliku.
3. Wykonano testy sprawdzające działanie algorytmu.
4. Program mierzy czas wykonania się algorytmu.
5. Kod opatrzono komentarzami.
6. Sporządzono schemat blokowy oraz pseudokod.

6. Appendix: kod programu

```
#include <iostream>

#include <ctime> //biblioteka potrzebna to losowania

#include <fstream> //biblioteka do wczytywania i zapisywania do txt

#include <chrono> //biblioteka do mierzenia czasu

using namespace std::chrono; //potrzebne do mierzenia czasu

using namespace std;

//Pierwsza funkcja służy do wczytania tabeli z pliku txt

int * WczytanieTablicy () { //Funkcja zdefiniowana tak żeby nie przyjmowała żadnych argumentów i tak żeby zwracała wczytaną tablicę

    fstream wczytywanie; //Tworzenie zmiennej, do której zapisany będzie plik, typu do tego przeznaczonego

    wczytywanie.open("Liczby.txt", ios::in); //Otwieranie wcześniej przygotowanego pliku tekstowego

    int i=0; //Deklaracja iteratora który będzie służył do zapisywania danych i równocześnie do zmierzenia długości tablicy

    int * WczytanaTabela = new int[i]; //Tworzenie tablicy dynamicznej do której zapisane będą liczby

    while(!wczytywanie.eof()) { //While wykonuje się aż skończy się plik

        wczytywanie >> WczytanaTabela[i]; //Po kolei wczytywane są liczby na kolejne miejsca w tablicy

        i++; //Iterator jest zwiększany przy każdym wczytaniu liczby, co na raz liczy ile liczb jest wczytane i służy do wskazania kolejnego miejsca na które wpisywana jest liczba

    }

    wczytywanie.close(); //Zamykanie pliku

    //Wypisywanie do konsoli wczytanej tablicy

    cout<<"Wczytana Tablica = [ ";

    for (int j=0; j<i; j++) { //For idący od 0 do i (które jest jest ilością liczb)
```

```

        cout <<WczytanaTabela[j]<<" ";
    }
    cout<<"]"<<endl;

return WczytanaTabela; //Funkcja zwraca wczytaną tablicy
}

//Druga funkcja generuje tablicę liczb naturalnych od -100 do 100
int * GenerowanieTablic (int N) { //Przyjmuje argument długości tabicy do wygenerowania
i zwraca tablicę

    srand((unsigned) time(0)); //Potrzebne do generowania liczb losowych
    int * GenerowanaTablica = new int[N]; //Deklaracja tablicy dynamicznej na liczby

    for (int j=0; j<N; j++){ //for działający N (przesłana do funkcji porządkana długość tablicy)
    razy
        GenerowanaTablica[j]=((rand()%400)-200); //Wykonywanie na rand modulo 200 daje
    liczby od 0 do 200, a potem odjęcie 100, daje liczby od -100 do 100
    }

    //Kolejna sekcja do wyświetlania, przeszkadza przy generowaniu dużej ilości tablic
    //cout<<"Wygenerowana Tablica = [ ";
    //for (int j=0; j<N; j++){
    //    cout <<GenerowanaTablica[j]<<" ";
    //}
    //cout<<"]"<<endl;

    return GenerowanaTablica; //Funkcja zwraca wygenerowaną tablicę
}

//Główna funkcja projektu, bierze tablicę, znajduje w niej liczby ujemne i przesuwa je na
koniec

```

```
int * GłównaFunkcjaPrzesuwania (int PrzesuwanaTablica[]){ //Funkcja przyjmuje jeden argument, tablicę do przesortowania, i zwraca przesortowaną tablicę
```

```
int N=sizeof(PrzesuwanaTablica); //Mierzenie długości tablicy
```

```
int TablicaUjemnych[N]; //Tworzenie pustej tablicy gdzie zapisane będą liczby ujemne
```

```
int IteratorUjemnych = 0; //Tworzenie iteratora który będzie służył do wstawiania do tablicy ujemnych i do liczenia ich ilości
```

```
int TablicaNieujemnych[N]; //Tworzenie pustej tablicy gdzie zapisane będą liczby nieujemne
```

```
int IteratorNieujemnych = 0; //Tworzenie iteratora który będzie służył do wstawiania do tablicy nieujemnych i do liczenia ich ilości
```

```
int * PrzesunietaTablica = new int[N]; //Tworzenie tablicy dynamicznej na finalny wynik
```

```
int IteratorPrzesunietejtablicy = 0; //Tworzenie iteratora który będzie służył do wstawiania do tablicy finalnej
```

```
for (int j=0; j<N; j++){ //For wykonuje się N (długość tablicy) razy
```

```
    if(PrzesuwanaTablica[j]<0){ //Jeśli element tablicy jest mniejszy od zera to przejście do pierwszej sekcji
```

```
        TablicaUjemnych[IteratorUjemnych]=PrzesuwanaTablica[j]; //Zapisywanie liczby do tablicy z ujemnymi
```

```
        IteratorUjemnych++; //Zwiększanie o 1 liczby ujemnych elementów i zarazem liczenie ich
```

```
    }else{ //Jeśli nie jest mniejszy od zera to do drugiej sekcji
```

```
        TablicaNieujemnych[IteratorNieujemnych]=PrzesuwanaTablica[j]; //Zapisywanie liczby do tablicy z nieujemnymi
```

```
        IteratorNieujemnych++; //Zwiększanie o 1 liczby nieujemnych elementów i zarazem liczenie ich
```

```
    }
```

```
}
```

```
for (int j=0; j<IteratorNieujemnych; j++){ //For wykonuje się tyle razy, ile znaleziono liczb nieujemnych
```

```
    PrzesunietaTablica[IteratorPrzesunietejtablicy]=TablicaNieujemnych[j]; //Dopisywanie do Finalnej tablicy liczb nieujemnych
```

```
    IteratorPrzesunietejtablicy++; //Zwiększanie iteratora elementów w finalnej tabeli i zarazem liczenie ich
```

```
}
```

```
    for (int j=0; j<IteratorUjemnych; j++){//For wykonuje się tyle razy, ile znaleziono liczb ujemnych
```

```
        PrzesunietaTablica[IteratorPrzesunietejtabelicy]=TablicaUjemnych[j]; //Dopisywanie do Finalnej tablicy liczb ujemnych (to dalej ten sama tablica z poprzedniego fora)
```

```
        IteratorPrzesunietejtabelicy++; //Zwiększanie iteratora elementów w finalnej tabeli i zarazem liczenie ich (to dalej ten sam iterator z poprzedniego fora)
```

```
    }
```

```
    return PrzesunietaTablica; //Funkcja zwraca przesuniętą tablicę
```

```
}
```

```
//Funkcja main, tu odpalane są inne funkcje i ich obsługa
```

```
int main()
```

```
{
```

```
    int * Tablica; //Tworzenie tablicy dynamicznej do której będą przypisywane wyniki
```

```
    Tablica = WczytanieTablicy (); //Wywołanie funkcji wczytywania tablicy i przypisanie wyjścia do zmiennej "Tablica" (czyli tablicy z pliku)
```

```
    Tablica = GłownaFunkcjaPrzesuwania(Tablica); //Wywołanie głównej funkcji gdzie argumentem jest tablica wczytana z pliku
```

```
    //Wypisanie wyniku przesuwania
```

```
    cout<<"Tablica po przesunieciu = [ ";
```

```
    for (int j=0; j<sizeof(Tablica); j++){
```

```
        cout<<Tablica[j]<<" ";
```

```
    }
```

```
    cout<<"]"<<endl;
```

```
    fstream zapisywanie; //Tworzenie zmiennej, która będzie przekazywać dane do pliku, typu do tego przeznaczonego
```

```
    zapisywanie.open("Wynik.txt", ios::out); //Otwieranie nowego pliku tekstowego do którego zapisywany będzie wynik
```

zapisywanie<<"Tablica po przesunieciu = ["; //Zamiast "cout" mamy zmienną "zapisywanie" więc wszystko idzie do pliku txt

```
for (int j=0; j<sizeof(Tablica); j++){
```

```
    zapisywanie<<Tablica[j]<<" ";
```

```
}
```

```
zapisywanie<<"]"<<endl;
```

```
zapisywanie.close(); //Zamknięcie pliku
```

fstream zapisywanie2; //Tworzenie zmiennej, która będzie przekazywać dane do pliku, typu do tego przeznaczonego

zapisywanie2.open("Czasy.txt", ios::out); //Otwieranie nowego pliku tekstowego do którego zapisywane będą wyniki

for (int ile=100000; ile<250000; ile+=100){ //For który wykonuje się 1500 razy, wartość iteratora jest na raz też długością tablicy którą każemy wygenerować programowi. Jest ona tak duża bo co dopiero taka długość daje jakikolwiek czas działania

Tablica = GenerowanieTablic (ile); //Wywołanie generowania tablic o długości zgodnej z iteratorem for'a

```
auto start = high_resolution_clock::now(); //Start mierzenia czasu
```

GlownaFunkcjaPrzesuwania(Tablica); //Wywołanie głównej funkcji z wygenerowaną tablicą

```
auto koniec = high_resolution_clock::now(); //Koniec mierzenia czasu
```

```
auto duration = duration_cast<microseconds>(koniec - start); //Odejmowanie czasu początkowego od końcowego i konwersja na microsekundy
```

```
zapisywanie2<<duration.count()<<endl; //zapisywanie każdego czasu do plik
```

```
}
```

```
zapisywanie.close(); //zamknięcie pliku
```

```
return 0;
```

```
}
```