

# **Zadanie projektowe nr 2**

## **Algorytmy i Struktury**

### **Danych**

**Inżynieria i analiza danych, I rok**

Kamil Hansel  
Numer indeksu: 166728  
Grupa: P03

## 1. Wstęp

**Sortowanie** – jeden z podstawowych problemów informatyki, polegający na uporządkowaniu zbioru danych względem pewnych cech charakterystycznych każdego elementu tego zbioru. Szczególnym przypadkiem jest sortowanie względem wartości każdego elementu, np. sortowanie liczb, słów itp.

Algorytmy sortujące, które dla elementów o tej samej wartości zachowują w tablicy końcowej kolejność tablicy wejściowej, nazywamy algorytmami **stabilnymi**.

<sup>1</sup> Źródło: <https://pl.wikipedia.org/wiki/Sortowanie>

## 2. Sortowanie przez wybór

Idea algorytmu **sortowania przez wybór** jest bardzo prosta. Załóżmy, iż chcemy posortować zbiór liczbowy rosnąco. Zatem element najmniejszy powinien znaleźć się na pierwszej pozycji. Szukamy w zbiorze **elementu najmniejszego** i wymieniamy go z **elementem na pierwszej pozycji**. W ten sposób element najmniejszy znajdzie się na swojej docelowej pozycji.

W identyczny sposób postępujemy z resztą elementów należących do zbioru. Znowu wyszukujemy element najmniejszy i zamieniamy go z elementem na drugiej pozycji. Otrzymamy dwa posortowane elementy. Procedurę kontynuujemy dla pozostałych elementów dotąd, aż wszystkie będą posortowane.

Sortowanie przez wybór należy do algorytmów niestabilnych a jego złożoność czasowa przedstawiona jest wzorem:  $O(n^2)$ .

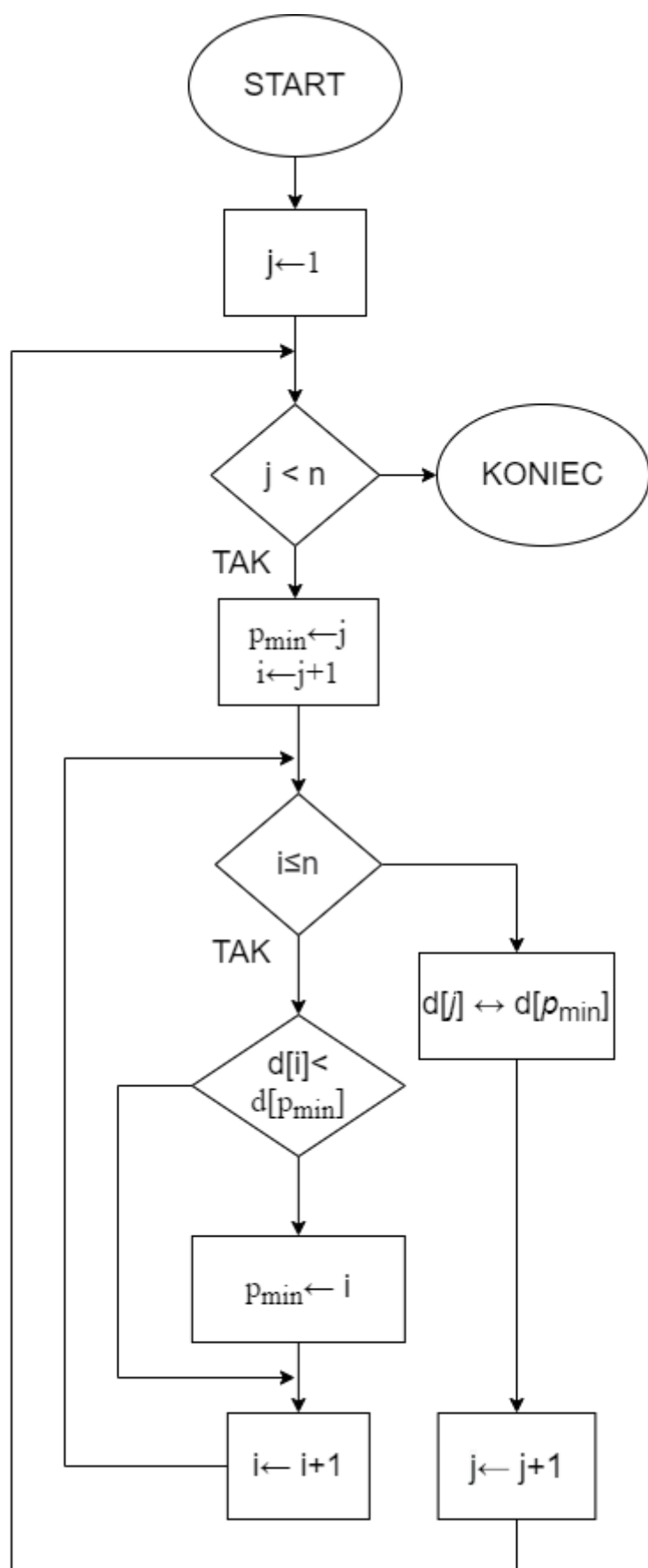
<sup>1</sup> Źródło: [https://eduinf.waw.pl/inf/alg/003\\_sort/0009.php](https://eduinf.waw.pl/inf/alg/003_sort/0009.php)  
[https://pl.wikipedia.org/wiki/Sortowanie\\_przez\\_wybieranie](https://pl.wikipedia.org/wiki/Sortowanie_przez_wybieranie)

Zbiór	Opis operacji
4 7 <b>2</b> 9 3	Wyszukujemy najmniejszy element w zbiorze. Jest nim liczba 2.
<b>2</b> 7 4 9 3	Znaleziony element minimalny wymieniamy z pierwszym elementem zbioru - liczbą 4
<b>2</b> 7 4 9 <b>3</b>	Wśród pozostałych elementów wyszukujemy element najmniejszy. Jest nim liczba 3.
<b>2</b> 3 4 9 7	Znaleziony element minimalny wymieniamy z drugim elementem zbioru - liczbą 7.
<b>2</b> 3 <b>4</b> 9 7	Znajdujemy kolejny element minimalny - liczbę 4.
<b>2</b> 3 4 9 7	Wymieniamy go z samym sobą - element ten nie zmienia zatem swojej pozycji w zbiorze.
<b>2</b> 3 4 9 <b>7</b>	Znajdujemy kolejny element minimalny
<b>2</b> 3 4 7 9	Wymieniamy go z liczbą 9
<b>2</b> 3 4 7 9	Ostatni element jest zawsze na właściwej pozycji. Sortowanie zakończone

Tabela obrazująca procedury sortowania przez wybór.

Źródło: [https://eduinf.waw.pl/inf/alg/003\\_sort/0009.php](https://eduinf.waw.pl/inf/alg/003_sort/0009.php)

## Schemat blokowy

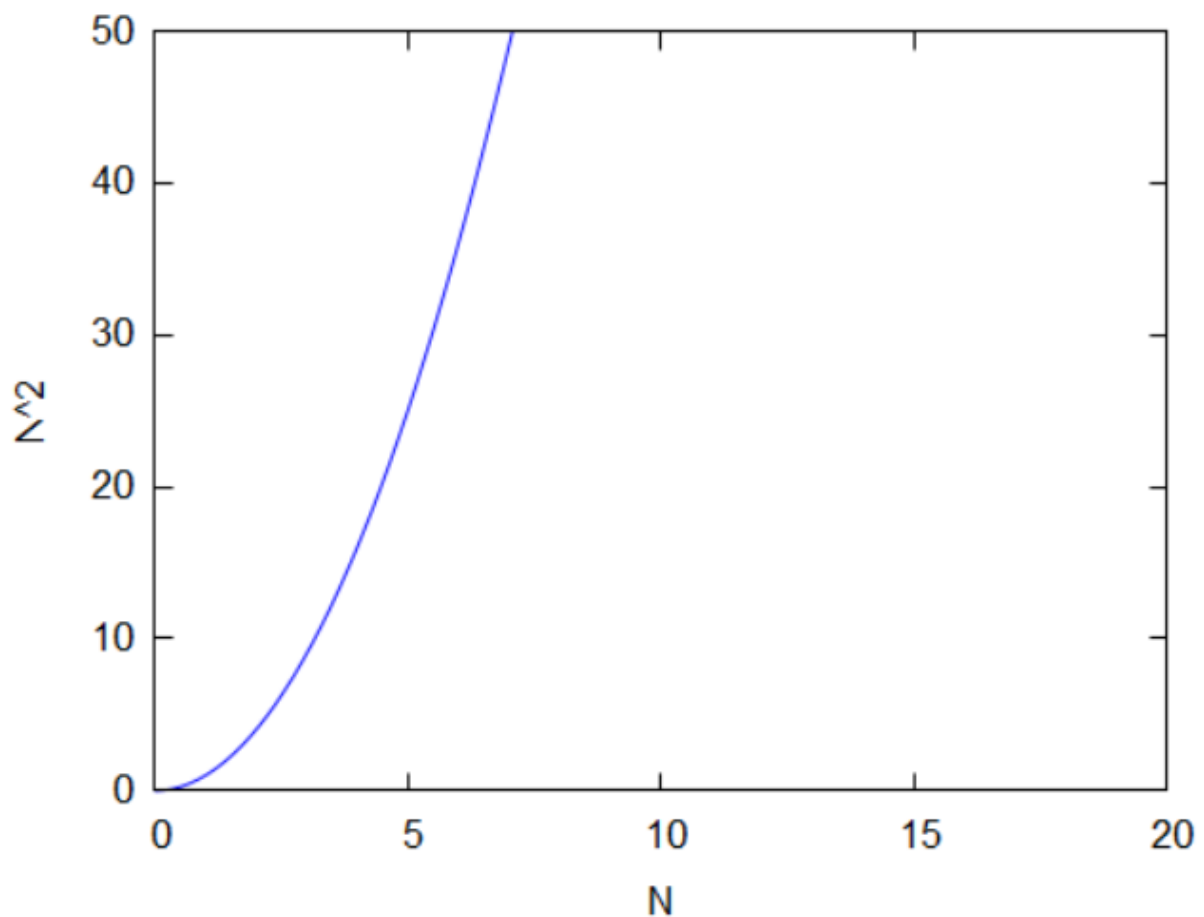


## Pseudokod

```
Sortuj L,n wykonaj
Dla i←0 i<n-1,i zwiększamy o 1
Min←i;
Dla int j←i+1,j<n;j zwiększamy o 1
Jeżeli L[j]<L[min] wykonaj
Min=j
Jeżeli i różne od min wykonaj
t←L[i]
L[i] ←L[min]
L[min] ←t
```

```
Przypadek 1
Fstream z1
Otwórz z1
Wypisz „podaj ilość wyrazów zbioru”
Wpisz n
L ←nowy [n]
Wypełnij pseudolosowymi liczbami
Dla i←0 i<n,i zwiększamy o 1 L ← pseudolosowe liczby
Sortuj L,n
Dla i←0,i<n i zwiększamy o 1
Wypisz L
Dla i←0, i<n i zwiększamy o 1
Zapisz L
Usuń L
Zatrzymaj system
Zamknij z1
```

Wykres przedstawiający złożoność czasową sortowania przez wybór



Kod sortowania przez wybór z możliwością odczytu z pliku oraz zapisu do pliku

```
7 void sort(double* L, int n){
8     for(int i = 0; i < n - 1; i++){
9         int min = i;
10        for(int j = i + 1; j < n; j++){
11            if(L[j] < L[min]){
12                min = j;
13            }
14        }
15        if(i != min){
16            double t = L[i];
17            L[i] = L[min];
18            L[min] = t;
19        }
20    }
21 }
```

```

48 | case 1: { //sortowanie przez wybor z losowymi elementami
49 |     ofstream z1; //wprowadzamy funkcje umożliwiające zapis oraz odczyt pliku
50 |     z1.open("wyniki sortowania przez wybor.txt",ios::out); //tworzymy plik tekstowy do zapisania posortowanej tablicy
51 |     int n;
52 |     cout<<"Podaj ilość wyrazów zbioru: "<<endl;
53 |     cin >> n;
54 |     double* L = new double[n];
55 |     srand((unsigned)time(NULL)); //wypełnianie tablicy pseudolosowymi wyrazami
56 |     for(int i = 0; i < n; i++)L[i] = rand()%1000000; //nadanie przedziałów pseudowylosowanym wyrazom
57 |     sort(L, n);
58 |     for(int i = 0; i < n; i++)
59 |         cout << L[i] << " ";
60 |     for(int i = 0; i < n; i++)
61 |         z1<<L[i]<<" "; //zapisujemy posortowaną tablicę do pliku tekstowego
62 |     delete[] L; //usunięcie tablicy
63 |     system("pause");
64 |     z1.close(); //zamknięcie strumienia po wczytaniu
65 |     return 0;
66 | }
67 |
68 | //sortowanie przez wybor z odczytem z pliku
69 | case 2: {
70 |     ifstream z2; //wprowadzamy funkcje umożliwiające zapis oraz odczyt pliku
71 |     z2.open("liczby sortowania przez wybor.txt",ios::in); //wczytanie danych z pliku
72 |     int n;
73 |     if(z2.good()==false)
74 |     {
75 |         cout<<"plik nie istnieje";
76 |         exit(0);
77 |     }
78 |     string linia;
79 |     int nr_linii=1;
80 |     while(getline(z2,linia))
81 |     {
82 |         switch(nr_linii)
83 |         {
84 |             case 1:n=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbę
85 |             default:
86 |                 nr_linii++;
87 |         }
88 |         double* L = new double[n];
89 |         srand((unsigned)time(NULL)); //wypełnianie tablicy pseudolosowymi wyrazami
90 |         for(int i = 0; i < n; i++)L[i] = rand()%1000000; //nadanie przedziałów pseudowylosowanym wyrazom
91 |         sort(L, n);
92 |         for(int i = 0; i < n; i++)
93 |             cout << L[i] << " ";
94 |         delete[] L; //usunięcie tablicy
95 |     }
96 | }
97 | }
98 | }
99 | }
100 | }
101 | }
102 | }
103 | }
104 | }
105 | }
106 | }
107 | }
108 | }
109 | }
110 | }
111 | }
112 | }
113 | }

```

Program wykonany w środowisku Code::Blocks IDE w języku C++

Wyniki programu

```
"C:\Users\Kamil\Desktop\lab1\projekt wyb\wyb\main.exe"
1 - sortowanie przez wybor z losowymi elementami
2 - sortowanie przez zliczanie z losowymi elementami
3 - sortowanie przez wybor z odczytem z pliku
4 - - sortowanie przez zliczanie z odczytem z pliku

Wybierz sposob dzialania programu: 1
Podaj ilosc wyrazow zbioru:
25
686 1861 2829 4581 7732 8953 9111 12223 13937 16616 16797 17286 17375 18427 21042 22502 23568 23794 23929 24464 27156 27445 27557 31555 31987 Press any key to continue . . .

Process returned 0 (0x0)   execution time : 6.751 s
Press any key to continue.

wyniki sortowania przez wybor — Notatnik
Plik  Edycja  Format  Widok  Pomoc
686 1861 2829 4581 7732 8953 9111 12223 13937 16616 16797 17286 17375 18427 21042 22502 23568 23794 23929 24464 27156 27445 27557 31555 31987
```

### 3. Sortowanie przez zliczanie

**Sortowanie przez zliczanie**(ang. *counting sort*) – metoda sortowania danych, która polega na sprawdzeniu ile wystąpień kluczy mniejszych od danego występuje w sortowanej tablicy

Algorytm zakłada, że klucze elementów należą do skończonego zbioru, co ogranicza możliwości jego zastosowania

Główną zaletą tej metody jest liniowa złożoność obliczeniowa algorytmu –  $O(n+k)$  ( $n$  – oznacza liczebność zbioru,  $k$  – rozpiętość danych, czyli w przypadku liczb

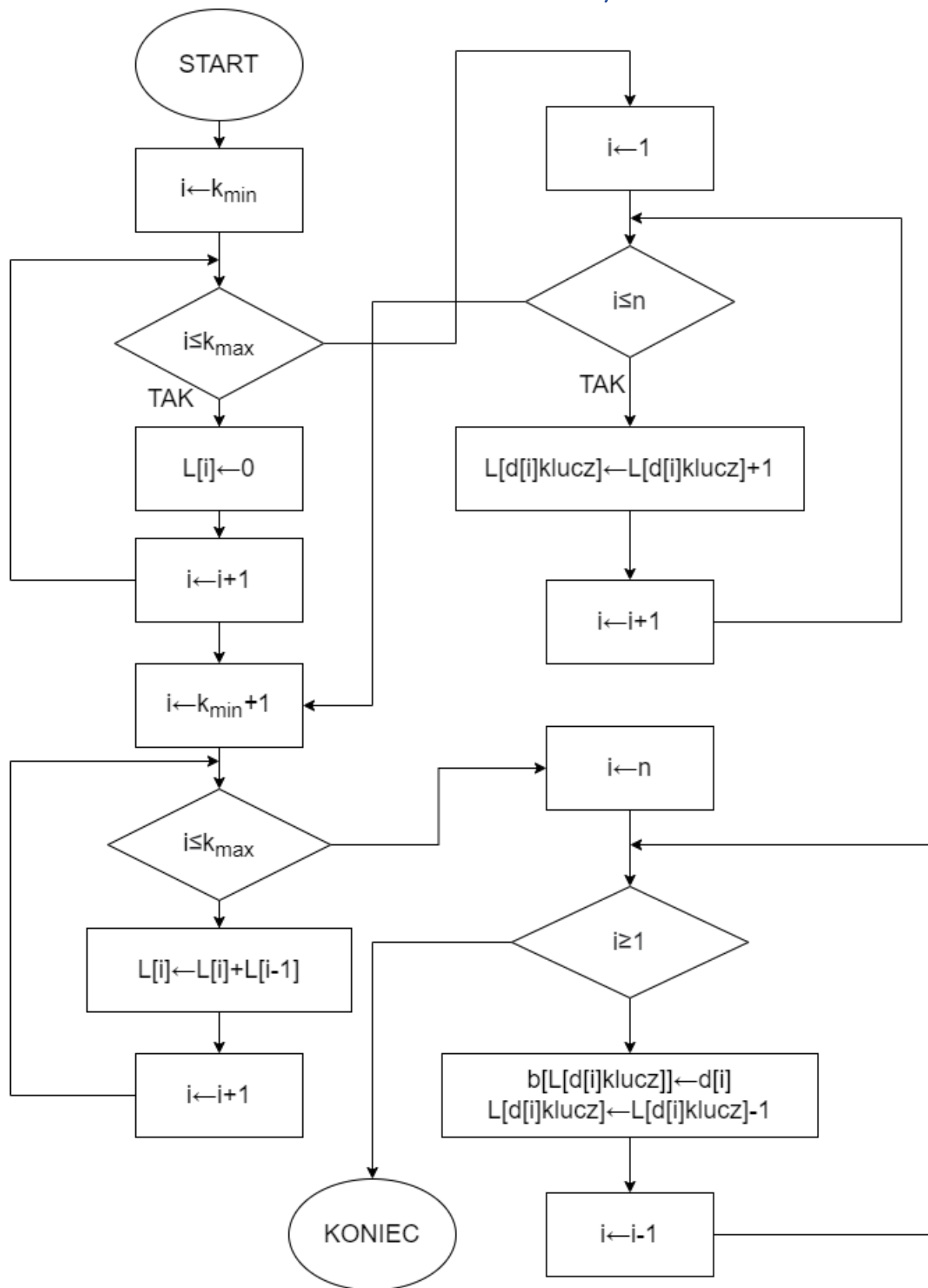


całkowitych: powiększoną o 1 różnicę między maksymalną a minimalną wartością, np. rozpiętość liczb w Dużym Lotku wynosi  $(49-1) + 1 = 49$ )

Największymi ograniczeniami algorytmu są konieczność uprzedniej znajomości zakresu danych i złożoność pamięciowa (wymaga dodatkowo  $O(k)$  lub  $O(n+k)$  pamięci)

Źródło: [https://pl.wikipedia.org/wiki/Sortowanie\\_przez\\_zliczanie](https://pl.wikipedia.org/wiki/Sortowanie_przez_zliczanie)

## Schemat blokowy



## Pseudokod

Sortuj L,n, lista, zakres od, zakres do, wykonaj

D1 ← zakres do – zakres od +1

Zlicz = nowy d1

Dla i ← 0 i < n-1, i zwiększamy o 1

Zlicz[i] ← 0

Dla int j ← i+1, j < n; j zwiększamy o 1

Zlicz[lista[i]-zakres od) zwiększamy o 1

x ← 0

dla int j ← i+1, j < n; j zwiększamy o 1

dla int j ← i+1, j < n; j zwiększamy o 1

lista[x zwiększamy o 1] ← i + zakres od

Przypadek 2

N, zakres od, zakres do

Fstream x1

Otwórz x1

Wypisz „podaj ilość wyrazów zbioru”

Wpisz n

Wypisz „podaj zakresy zbioru”

Wpisz zakres od, zakres do

Wypełnij pseudolosowymi liczbami

L ← nowy L

Dla i ← 0 i < n, i zwiększamy o 1 L ← pseudolosowe liczby

Sortuj L, zakres od, zakres do, n

Dla i ← 0, i < n i zwiększamy o 1

Wypisz L

Dla i ← 0, i < n i zwiększamy o 1

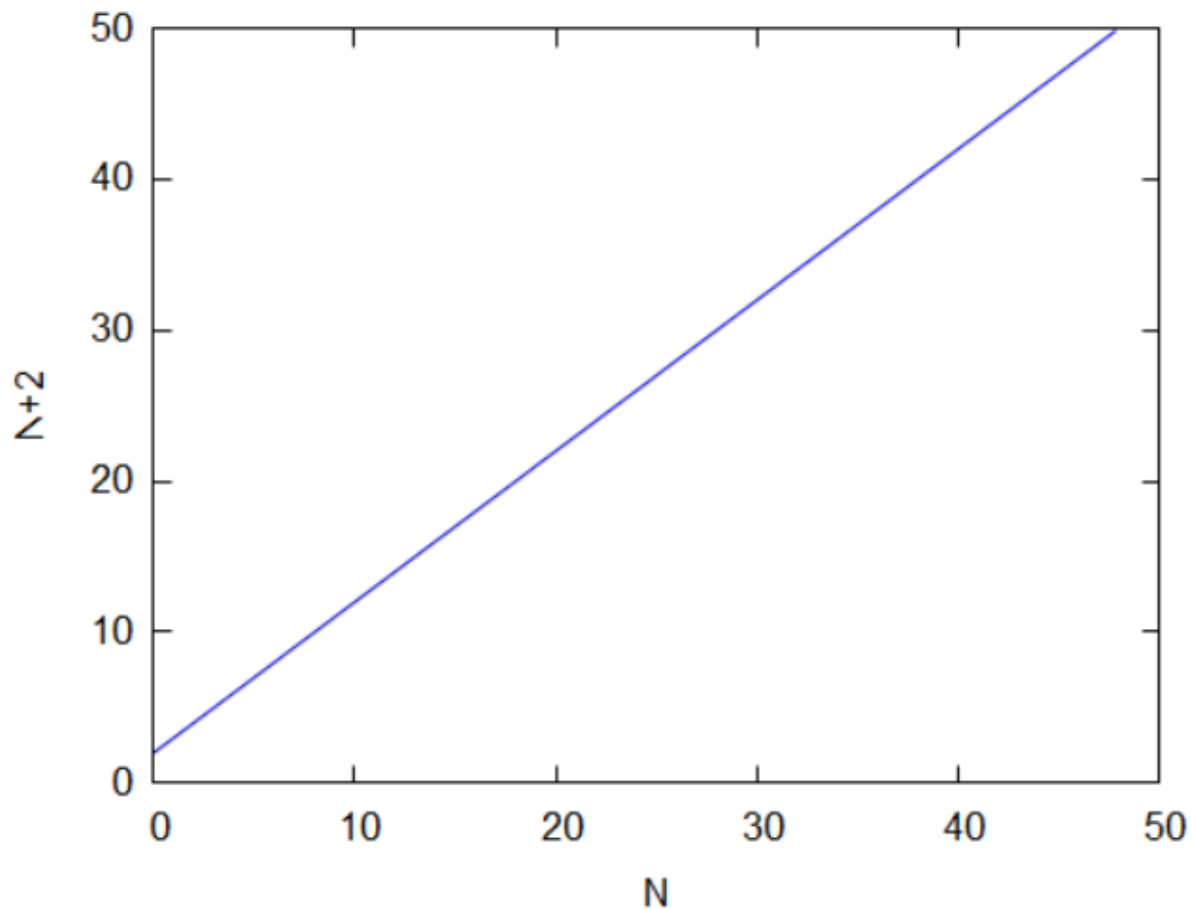
Zapisz L

Usuń L

Zatrzymaj system

Zamknij x1

## Wykres przedstawiający złożoność czasową sortowania przez zliczanie



## Kod sortowania przez zliczanie z możliwością odczytu z pliku oraz zapisu do pliku

```
22 void sort(int* lista, int zakres_od, int zakres_do, int n){
23     int dl = zakres_do - zakres_od + 1;
24     int* zlicz = new int [dl];
25     for(int i = 0; i < dl; i++)
26         zlicz[i] = 0;
27     for(int i = 0; i < n; i++)
28         zlicz[lista[i] - zakres_od]++;
29     int x = 0;
30     for(int i = 0; i < dl; i++)
31         for(int j = 0; j < zlicz[i]; j++)
32             lista[x++] = i + zakres_od;
33 }
67 case 2: { //sortowanie przez zliczanie z losowymi elementami
68     int n, zakres_od, zakres_do;
69     fstream xl; //wprowadzamy funkcje umożliwiające zapis oraz odczyt pliku
70     xl.open("wyniki sortowania przez zliczanie.txt", ios::out); //tworzymy plik tekstowy do zapisania posortowanej tablicy
71     cout<<"Podaj ilość wyrazów zbioru: "<<endl;
72     cin >> n;
73     cout<<"podaj zakresy zbioru: "<<endl;
74     cin >> zakres_od >> zakres_do;
75     srand((unsigned)time(NULL));
76     int* L = new int[n];
77     for(int i = 0; i < n; i++)L[i] = zakres_od + rand() % (zakres_do-zakres_od+1); //nadanie przedziałów pseudolosowanych wyrazom
78     sort(L, zakres_od, zakres_do, n);
79     for(int i = 0; i < n; i++)
80         cout << L[i] << " ";
81     for(int i = 0; i < n; i++)
82         xl<<L[i]<<" "; //zapisujemy posortowaną tablicę do pliku tekstowego
83     delete[] L; //usuwanie tablicy
84     system("pause");
85     xl.close(); //zamykanie strumienia po wczytaniu
86     return 0;
87 }
```

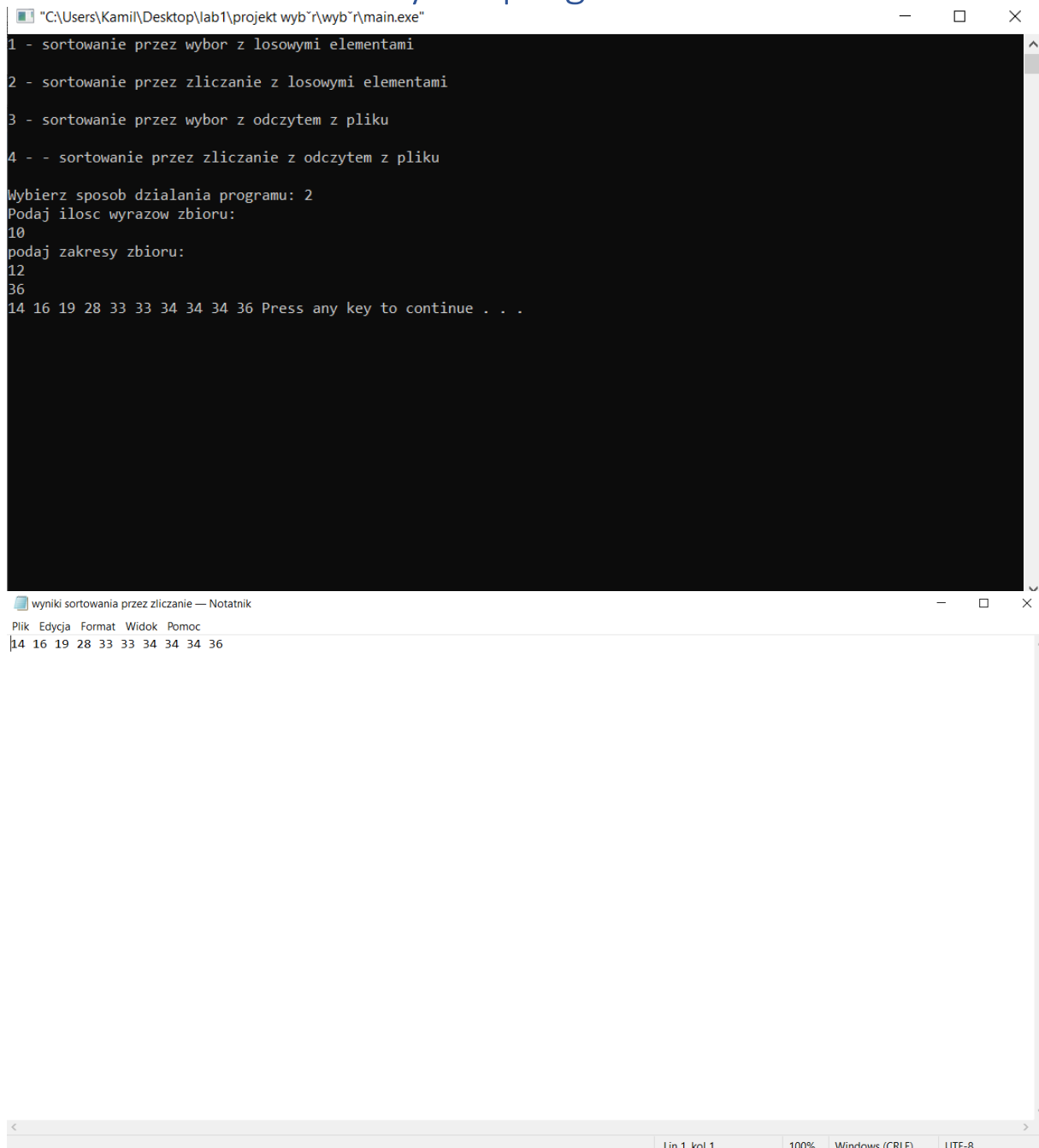
```

118 | case 4: //sortowanie przez zliczanie z odczytem z pliku
119 | {
120 |     int n, zakres_od, zakres_do;
121 |     fstream x2; //składamy funkcję umożliwiającą zapis oraz odczyt pliku
122 |     x2.open("liczby sortowania przez zliczanie.txt", ios::in); // wczytywanie danych z pliku
123 |     if(x2.good()==false)
124 |     {
125 |         cout<<"plik nie istnieje"; //w przypadku nie istnienia pliku wywielila sie komunikat
126 |         exit(0);
127 |     }
128 |     string linia;
129 |     int nr_linii=1;
130 |     while(getline(x2, linia))
131 |     {
132 |         switch(nr_linii)
133 |         {
134 |             case 1:n=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbe
135 |             case 2:zakres_od=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbe
136 |             case 3:zakres_do=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbe
137 |         }
138 |         nr_linii++;
139 |     }
140 |     srand((unsigned)time(NULL)); //wypełnianie tablicy pseudolosowymi wyrazami
141 |     int* L = new int[n];
142 |     for(int i = 0; i < n; i++)L[i]=zakres_od + rand() % (zakres_do-zakres_od+1); //nadanie przedzialow pseudowylosowanym wyrazom
143 |     sort(L, zakres_od, zakres_do, n);
144 |     {
145 |         switch(nr_linii)
146 |         {
147 |             case 1:n=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbe
148 |             case 2:zakres_od=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbe
149 |             case 3:zakres_do=atoi(linia.c_str());break; //przekształcenie wyrazu zapisanego w pliku na liczbe
150 |         }
151 |         nr_linii++;
152 |     }
153 |     srand((unsigned)time(NULL)); //wypełnianie tablicy pseudolosowymi wyrazami
154 |     int* L = new int[n];
155 |     for(int i = 0; i < n; i++)L[i]=zakres_od + rand() % (zakres_do-zakres_od+1); //nadanie przedzialow pseudowylosowanym wyrazom
156 |     sort(L, zakres_od, zakres_do, n);
157 |     for(int i = 0; i < n; i++)
158 |     {
159 |         cout << L[i] << " ";
160 |     }
161 |     for(int i = 0; i < n; i++)
162 |     {
163 |         x2<<L[i]<<" ";
164 |     }
165 |     delete[] L; //usuniecie tablicy
166 |     system("pause");
167 |     x2.close(); // zamkniecie strumienia po wczytaniu
168 |     return 0;
169 | }
170 | }

```

**Program wykonany w środowisku Code::Blocks IDE w języku C++**

## Wyniki programu



The image shows a Windows desktop with two windows. The top window is a command prompt titled "C:\Users\Kamil\Desktop\lab1\projekt wyb\ryb\ryb\main.exe". It displays a menu with four options: 1 - sortowanie przez wybor z losowymi elementami, 2 - sortowanie przez zliczanie z losowymi elementami, 3 - sortowanie przez wybor z odczytem z pliku, and 4 - sortowanie przez zliczanie z odczytem z pliku. The user has selected option 2. The prompt then asks for the number of elements in the set, with '10' entered. It then asks for the range of the set, with '12' and '36' entered on separate lines. The final output is a list of numbers: 14 16 19 28 33 33 34 34 34 36, followed by the text 'Press any key to continue . . .'. The bottom window is a Notepad application titled 'wyniki sortowania przez zliczanie — Notatnik'. It contains the same list of numbers: 14 16 19 28 33 33 34 34 34 36. The Notepad window has a menu bar with 'Plik', 'Edycja', 'Format', 'Widok', and 'Pomoc'. The status bar at the bottom of the Notepad window shows 'Lin 1 kół 1', '100%', 'Windows (CP1250)', and 'UTF-8'.

```
"C:\Users\Kamil\Desktop\lab1\projekt wyb\ryb\ryb\main.exe"
1 - sortowanie przez wybor z losowymi elementami
2 - sortowanie przez zliczanie z losowymi elementami
3 - sortowanie przez wybor z odczytem z pliku
4 - - sortowanie przez zliczanie z odczytem z pliku

Wybierz sposob dzialania programu: 2
Podaj ilosc wyrazow zbioru:
10
podaj zakresy zbioru:
12
36
14 16 19 28 33 33 34 34 34 36 Press any key to continue . . .

wyniki sortowania przez zliczanie — Notatnik
Plik  Edycja  Format  Widok  Pomoc
14 16 19 28 33 33 34 34 34 36

Lin 1 kół 1  100%  Windows (CP1250)  UTF-8
```