Wykład 1_2
Algorytmy sortowania tablic
Sortowanie bąbelkowe

Elementy języka stosowanego do opisu algorytmu

Elementy	Poziom koncepcji	Poziom projektu
I. Struktury sterujące 1.bezpośrednie następstwo (A,B-czynności)	(1) wykonaj A, (2) a potem wykonaj B	(1) wykonaj A (2) wykonaj B
0. wybór warunkowy lub rozgałęzienie warunkowe (Q-warunek;A,B-czynności)	jeśli Q, to wykonaj A, jeśli Q, to wykonaj A, w przeciwnym razie wykonaj B	jeśli Q, to A jeśli Q to A, w przeciwnym razie B;
0. iteracja ograniczona (A - czynność; N - pewna liczba)	wykonaj A [dokładnie] N razy	wykonaj, co następuje, [dokładnie] N razy
0. iteracja warunkowa (Q - warunek, A - czynność X, N - pewne liczby)	wykonuj A aż do Q lub dopóki Q, wykonuj A	wykonuj, co następuje, aż X < N dopóki X< N, wykonuj, co następuje
0. składanie struktur sterujących np. iteracje zagnieżdżone czyli pętle zagnieżdżone (N, M - pewne liczby, A - czynność)	(1) wykonaj , co następuje, [dokładnie] N razy: (1.0) wykonaj A [dokładnie] M razy	(0) wykonuj, co następuje, [dokładnie] N razy: (1.0) wykonaj, co następuje, [dokładnie] M razy: (1.1.1).....
II. Struktury danych: 0. zmienne zawierające jedna daną (A - nazwa zmiennej ←wstawianie danej z prawej strony strzałki do zmiennej po lewej stronie strzałki		A← 1
2.tablice jednowymiarowe (T - nazwa tablicy; X - numer elementu)		T (X) - element tablicy o numerze X,
3. tablice dwuwymiarowe (T - nazwa tablicy X - numer wiersza Y - numer kolumny)		T (X, Y) - element tablicy T o numerze wiersza X i numerze kolumny Y

1.Ogólne sformułowanie algorytmu sortowania bąbelkowego - poziom koncepcji

Algorytm sortowania N elementów:

(1) wykonaj, co następuje, N -1 razy
(1.1) wskaż na pierwszy element;
(1.2) wykonaj, co następuje, N - 1 razy:
(1.2.1) porównaj wskazany element z elementem następnym
(1.2.2) jeśli porównywane elementy są w niewłaściwej kolejności, zamień je miejscami;
(1.2.3) wskaż na następny element.

2. Uściślenie algorytmu sortowania bąbelkowego - poziom projektu

Algorytm sortowania N elementów:

(1) i ← 1;
(2) dopóki i <= N -1, wykonuj co następuje:
(2.1) j ← 1;
(2.2) dopóki j <= N - i, wykonuj, co następuje:
(2.2.1) jeśli T(j + 1) < T(j), to zamień je;
(2.2.2) j ← j + 1;
(2.3) i ← i +1;

Uwagi:

1.W algorytmie sortowania występują dwie pętle: pętla zewnętrzna ***(2),*** pętla wewnętrzna ***(2.2).***

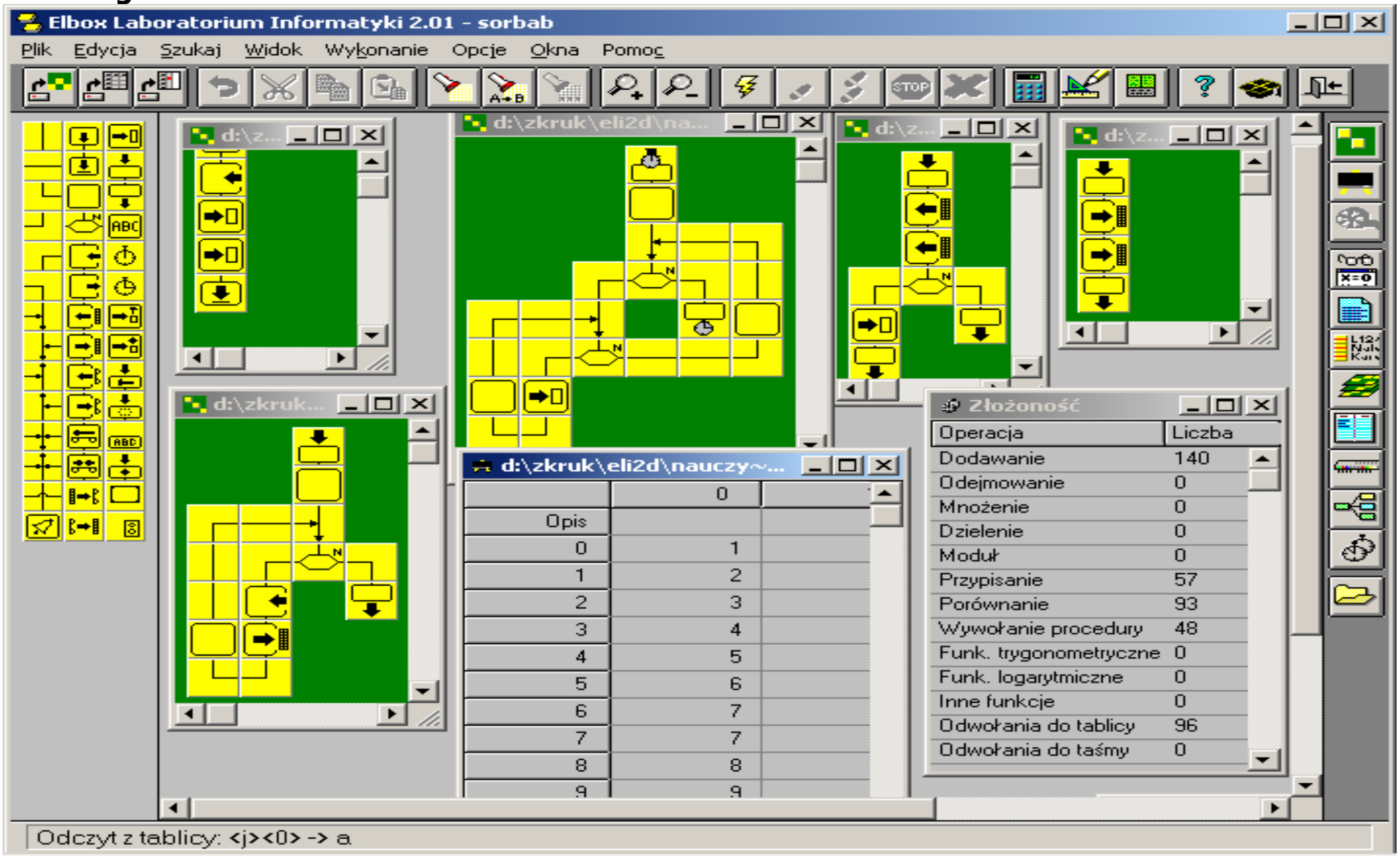
2. Powtórzenia działań w pętlach są numerowane następująco:
   - pętla ***(2)*** za pomocą zmiennej od ***i =1*** do ***N -1***
   - pętla ***(2.2)*** za pomocą zmiennej ***j =1*** do ***N - i.***

Lp	Numery elementów tablicy *T*							
i /j	1	2	3	4	5	6	7	8
1/2	Francja	**Grecja**	**Albania**	Egipt	Cypr	Hiszpania	Belgia	Dania
1/3	Francja	Albania	**Grecja**	**Egipt**	Cypr	Hiszpania	Belgia	Dania
1/4	Francja	Albania	Egipt	**Grecja**	**Cypr**	Hiszpania	Belgia	Dania
1/6	Francja	Albania	Egipt	Cypr	Grecja	**Hiszpania**	**Belgia**	Dania
1/7	Francja	Albania	Egipt	Cypr	Grecja	Belgia	**Hiszpania**	**Dania**
	Francja	*Albania*	*Egipt*	*Cypr*	*Grecja*	*Belgia*	*Dania*	***Hiszpania***
2/1	**Francja**	**Albania**	Egipt	Cypr	Grecja	Belgia	Dania	***Hiszpania***
2/2	Albania	**Francja**	**Egipt**	Cypr	Grecja	Belgia	Dania	***Hiszpania***
2/3	Albania	Egipt	**Francja**	**Cypr**	Grecja	Belgia	Dania	***Hiszpania***
2/5	Albania	Egipt	Cypr	Francja	**Grecja**	**Belgia**	Dania	***Hiszpania***
2/6	Albania	Egipt	Cypr	Francja	Belgia	**Grecja**	**Dania**	***Hiszpania***
	Albania	*Egipt*	*Cypr*	*Francja*	*Belgia*	*Dania*	***Grecja***	***Hiszpania***
3/2	Albania	**Egipt**	**Cypr**	Francja	Belgia	Dania	***Grecja***	***Hiszpania***
3/4	Albania	Cypr	Egipt	**Francja**	**Belgia**	Dania	***Grecja***	***Hiszpania***
3/5	Albania	Cypr	Egipt	Belgia	**Francja**	**Dania**	***Grecja***	***Hiszpania***
	Albania	*Cypr*	*Egipt*	*Belgia*	*Dania*	***Francja***	***Grecja***	***Hiszpania***
4/3	Albania	Cypr	**Egipt**	**Belgia**	Dania	***Francja***	***Grecja***	***Hiszpania***
4/4	Albania	Cypr	Belgia	**Egipt**	**Dania**	***Francja***	***Grecja***	***Hiszpania***
	Albania	*Cypr*	*Belgia*	*Dania*	***Egipt***	***Francja***	***Grecja***	***Hiszpania***
5/2	Albania	**Cypr**	**Belgia**	Dania	***Egipt***	***Francja***	***Grecja***	***Hiszpania***
	Albania	*Belgia*	*Cypr*	***Dania***	***Egipt***	***Francja***	***Grecja***	***Hiszpania***

Tab.1. Sortowanie bąbelkowe łańcuchów

Uwaga: Kolumna *Lp* zawiera numery stanów pętli zewnętrznej ***i*** oraz wewnętrznej ***j***, natomiast każdy wiersz odpowiada zawartości elementów tablicy ***T*** w stanie ***i/j*** przed zamianą elementów ***j*** z ***j+1***

3. Realizacja
3.1. Program ELI2D



3.2. Realizacja w C/C++

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

typedef int element;
const long N=20000L;
const int m=10;

inline void zamien(element &a, element &b);
inline void porownaj_zamien(element &a, element &b);
void wypelnij(element t[], long& ile);
void wyswietl(element t[], long ile);
void babelki(element t[], long l, long p);

void main()
{  element * t=new element[N];
    long ile=0;

   wypelnij(t,ile);       // ile po zakończeniu funkcji jest równe N
   babelki(t,0,ile-1);  // ile-1 jest równe N-1; 0 jest indeksem pierwszego elementu
                         //tablicy, ile- jest indeksem ostatniego elementu tablicy
   wyswietl(t,ile);
   getch();
}
inline void zamien(element  &a, element  &b)
      {     element  pom = a;
            a=b;
            b=pom;
      }
inline void porownaj_zamien(element &a, element &b)
      {     if (b<a) zamien(a,b);  }

void babelki(element t[], long l, long p)
      { for( long i =l; i<p; i++)
          for (long j=l;j<p-i; j++)
             porownaj_zamien(t[j], t[j+1]);
      }
```

```
void wypelnij(element t[], long& ile)
  {   srand(3);
      for (long i=0; i<N; i++)
            t[i]=rand();
      ile=N;
  }
void wyswietl(element t[], long ile)
{ for (long i=0; i<ile; i++)
        { printf("%d \n", t[i]);
          if (i%20==0)
            {  char z=getch();
               if (z=='k') return; }
        }
     printf("%ld \n", ile);
}
```

4. Badanie czasochłonności algorytmu

Analiza algorytmów polega na określeniu zasobów, jakie są potrzebne do ich wykonania.

Zasobami są:
- czas wykonania
- pamięć
- szerokość kanału komunikacyjnego
- układy logiczne.

Modelem obliczeń jest jednoprocesorowa maszyna z dostępem swobodnym do pamięci (RAM - Random Access Machine). Algorytmy są realizowane jako programy komputerowe, a instrukcje są wykonywane sekwencyjnie.

Czas wykonania jest zależny od **rozmiaru danych wejściowych**.

Rozmiar danych wejściowych jest zależny od związanego z nimi rozważanego problemu np.:
- sortowanie - liczba danych do posortowania
- mnożenie dwóch liczb - liczba bitów do reprezentowania danych w postaci binarnej
- problemy z danymi w postaci grafów - liczby wierzchołków i krawędzi.

Czas działania algorytmu dla konkretnych danych wejściowych jest wyrażony liczbą wykonanych prostych (elementarnych) operacji lub „kroków".
Zakłada się, że operacja elementarna jest maszynowo niezależna. Czas wykonania i-tego wiersza programu wymaga czasu c_j.

Analiza algorytmu sortowania bąbelkowego

Lp		Koszt	Liczba wykonań	
1	**for(long i =0; i<N-1; i++)**	c_1	N	
2	**for (long j=0;j<N-1-i; j++)**	c_2	$t_j = N - i+1;$	$\sum_{i=1}^{N-1}(N-i+1)$
3	**porownaj_zamien(t[j],t[j+1]);**	c_3	$t_{j-1}=N-i+1-1=N-i;$	$\sum_{i=1}^{N-1}(N-i\}$
4	**if (b<a)**	c_4	$t_{j-1}=N-i+1-1=N-i;$	$\sum_{i=1}^{N-1}(N-i\}$
5	**zamien(a,b);**	c_5	$t_{j-1}=N-i+1-1=N-i;$	$\sum_{i=1}^{N-1}(N-i\}$
6	**{ element pom=a;**	c_6	$t_{j-1}=N-i+1-1=N-i;$	$\sum_{i=1}^{N-1}(N-i\}$
7	a=b;	c_7	$t_{j-1}=N-i+1-1=N-i;$	$\sum_{i=1}^{N-1}(N-i\}$
8	b=pom; }	c_8	$t_{j-1}=N-i+1-1=N-i;$	$\sum_{i=1}^{N-1}(N-i\}$

$$T(N)=c_1N+ c_2\sum_{i=1}^{N-1}(N-i+1) +$$

$$c_3\sum_{i=1}^{N-1}(N-i) +c_4\sum_{i=1}^{N-1}(N-i) +c_5\sum_{i=1}^{N-1}(N-i) +c_6\sum_{i=1}^{N-1}(N-i) +c_7\sum_{i=1}^{N-1}(N-i) +c_8\sum_{i=1}^{N-1}(N-i) =$$

$$= c_1N + c_2(\frac{N(N+1)}{2}-1) + (c_3 + c_4 + c_5 + c_6+ c_7+ c_8)\ (\frac{N(N+1)}{2}-N) =$$

$$= c_1N + c_2\frac{1}{2}(N^2 + N) - c_2+ (c_3 + c_4 + c_5 + c_6+ c_7+ c_8)\frac{1}{2}(N^2 - N) =$$

$$= N^2\ \frac{1}{2}\ (c_2 + c_3 + c_4 + c_5 + c_6+ c_7+ c_8) +$$

$$N\ (c_1 + \frac{1}{2}\ c_2 - \frac{1}{2}\ c_3 - \frac{1}{2}\ c_4 - \frac{1}{2}c_5 - \frac{1}{2}c_6 - \frac{1}{2}c_7 - \frac{1}{2}c_8) - c_2$$

$$T(N) \approx c'' N^2 + c'''N$$

Wniosek: Dla dużych **N** czas wykonania algorytmu sortowania bąbelkowego zależy od kwadratu liczby danych.