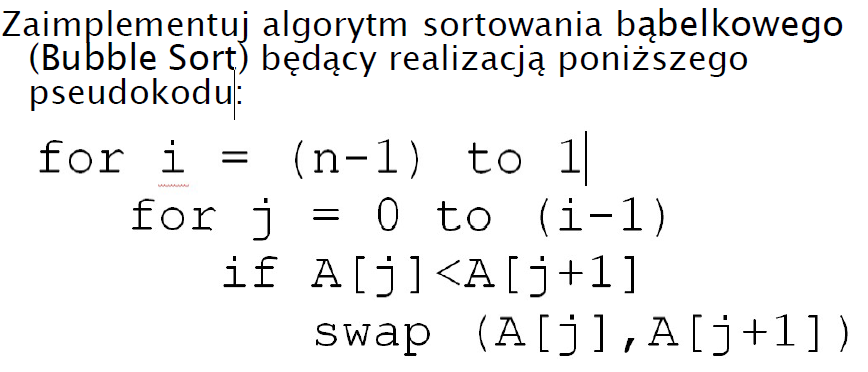
**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 7.1 |
| **Autor** | Mateusz Jasiński | **Grupa** | WCY20IY2S1 |
| **Temat** | Algorytmy sortowania | | |

1. Treść



* 1. Metoda realizacji

Przejść tablice o wielkości n dokładnie n-1 razy, porównując sąsiadujące ze sobą elementy.

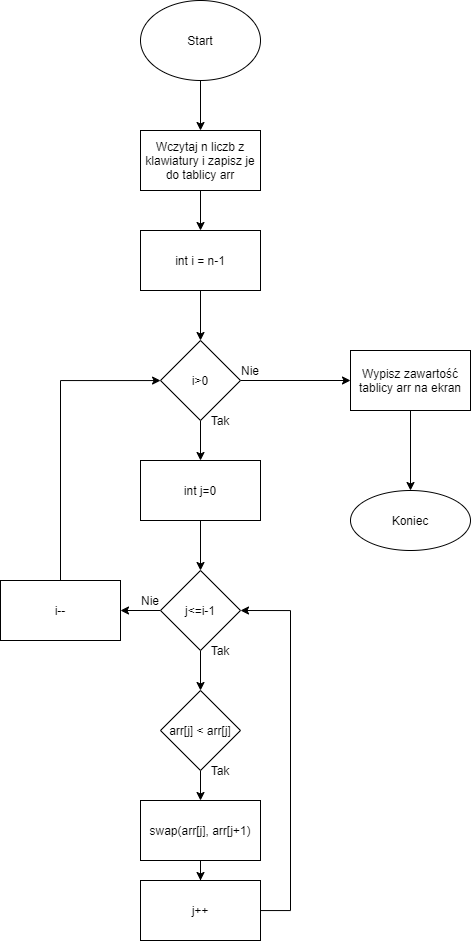
* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe

Ilość elementów i przedział wartości liczb – wprowadzone z klawiatury

* + 1. Dane wyjściowe

Posortowane dane – wyprowadzone na ekran

1. Realizacja
   1. Algorytm



* 1. Kod źródłowy

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int losuj(int a, int b) {

if(a>b) {

int temp=a;

a=b;

b=temp;

}

return rand()%(b-a+1)+a;

}

bubble\_sort(int \*tab, int n) {

int i, j, temp;

for(i=n-1;i>=1;i--)

for(j=0;j<=i-1;j++)

if(tab[j]>tab[j+1]) {

temp=tab[j];

tab[j]=tab[j+1];

tab[j+1]=temp;

}

}

main() {

srand(time(0));

int i, n, \*tab, a, b;

printf("Podaj ilosc elementow: ");

scanf("%d",&n);

tab=(int\*)malloc(n\*sizeof(int));

printf("Podaj przedzial wartosci elementow: ");

scanf("%d %d",&a,&b);

for(i=0; i<n; i++)

tab[i]=losuj(a,b);

bubble\_sort(tab, n);

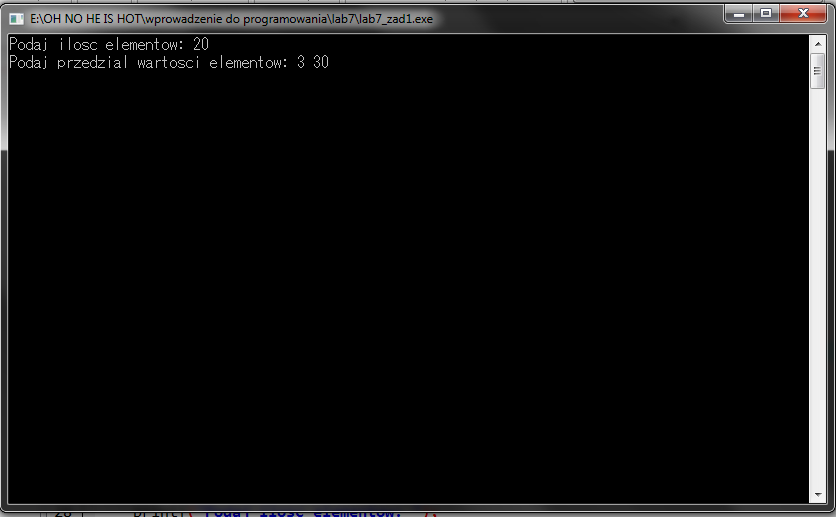
printf("Posortowana tablica:\n");

for(i=0; i<n; i++)

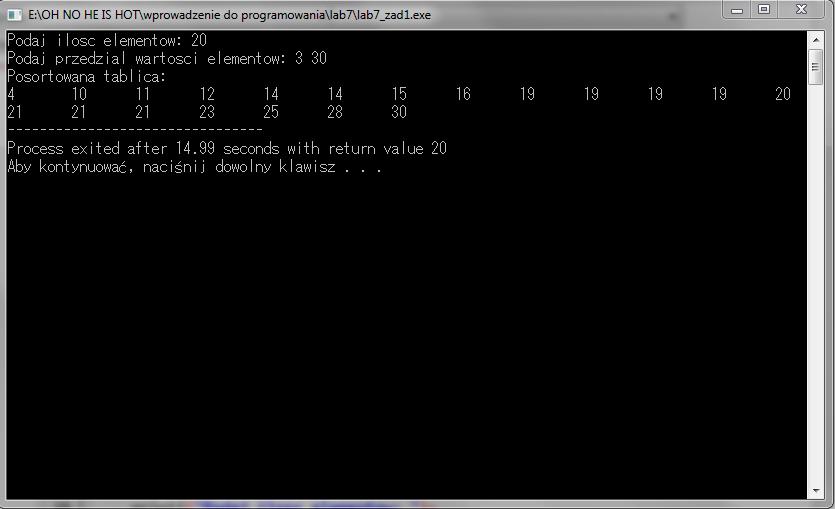
printf("%d\t",tab[i]);

}

* 1. Dane wejściowe



* 1. Dane wyjściowe



1. Wnioski

Złożoność obliczeniowa:

O(n^2) – sortowanie bąbelkowe przechodzi przez tablice n-1 i za każdym razem dokonuje n-1 porównań elementów, czyli wykonuje (n-1)^2 operacji.