PORÓWNANIE ALGORYTMÓW SORTUJĄCYCH

Działanie programu:

- po wczytaniu ilości losowanych liczb całkowitych ‘n’ wylosowana zostaje ‘n’-elementowa tablica
- stworzenie funkcji sortowania bąbelkowego
- stworzenie funkcji sortowania quicksort
- zmierzenie czasu sortowania bąbelkowego i quicksort w celu ich porównania

Kod źródłowy programu:

```
1   //booblesort vs. quicksort
2   #include <iostream>
3   #include <ctime>
4   #include <windows.h>
5
6   using namespace std;
7
8   clock_t start, stop;
9   double czas;
10
11  void buublesort(int *tab, int n){
12      for(int i=1;i<n;i++){
13          for(int j=n-1;j>0;j--){
14              if(tab[j]<tab[j-1]){
15                  int buff=tab[j-1];
16                  tab[j-1]=tab[j];
17                  tab[j]=buff;
18              }
19          }
20      }
21  }
```

```
22  void quicksort(int *tab, int lewy, int prawy){
23      int v=tab[(lewy+prawy)/2];
24      int i,j,x;
25      i=lewy;
26      j=prawy;
27      do{
28          while(tab[i]<v) i++;
29          while(tab[j]>v) j--;
30          if(i<=j)
31          {
32              x=tab[i];
33              tab[i]=tab[j];
34              tab[j]=x;
35              i++;
36              j--;
37          }
38      } while(i<=j);
39      if(j>lewy){
40          quicksort(tab,lewy, j);
41      }
42      if(i<prawy){
43          quicksort(tab, i, prawy);
44      }
45  }
```

```
46  int main(){
47      cout<<"Podaj ilosc elementow do posortowania"<<endl;
48      int n;
49      cin>>n;
50      int *tab;
51      int *tab2;
52      tab = new int[n];
53      tab2 = new int[n];
54      srand(time(NULL));
55  //generator nie posortowanej tablicy
56      //cout << "Tablica przed sortowaniem:" << endl;
57      for (int i=0;i<n;i++){
58          tab[i]=rand()%100000;
59          tab2[i]=tab[i];
60          //cout << tab[i]<< " ";
61      }
62      cout<<endl;
```

```
63  //sortowanie bombelkowe z pomiarem czasu
64      cout<<"Sortowanie bombelkowe trwa!!! "<<endl;
65      start = clock();
66      buublesort(tab,n);
67      stop = clock();
68      czas = (double)(stop-start)/CLOCKS_PER_SEC;
69      /*cout << "Tablica po posortowaniu:" << endl;
70      for (int i=0;i<n;i++){
71          cout << tab[i]<< " ";
72      }*/
73      cout<<endl;
74      cout<<"Czas: "<<czas<<" s"<<endl;
75  //sortowanie quicksort z pomiarem czasu
76      cout<<"Sortowanie quicksort trwa!!! "<<endl;
77      start = clock();
78      quicksort(tab2,0,n-1);
79      stop = clock();
80      czas = (double)(stop-start)/CLOCKS_PER_SEC;
81      cout << "Tablica po posortowaniu:" << endl;
82      /*for (int i=0;i<n;i++){
83          cout << tab2[i]<< " ";
84      }*/
85      cout<<endl;
86      cout<<"Czas: "<<czas<<" s"<<endl;
87      delete [] tab;//usuwa tablicê z pamieci komputera
88      delete [] tab2;
89      return 0;
90  }
```

Różnicę widać dopiero przy większej ilości danych do posortowania (np. 50 000 liczb).