```cpp
#include <iostream>
#include <ctime>
#include <time.h>
#include <cstdlib>
#include <fstream>
#include <cstring>
#include <conio.h>
#include <iomanip>

using namespace std;

int R, sposob, nr_dzialania; // zmienne potrzebne do prawidlowego dzialania rozmiaru tablicy, wyboru algorytmu sortujacego oraz poslugiwania sie menu programu
long int N; // zmienna potrzebna do okreslenia przedzialu losowanych liczb
clock_t start, stop; // zmienne start i stop do przechowywania cykli zegara
double czas; // zmienna czas typu double potrzebna do okreslenia czasu dzialania algorytmu w sekundach
ofstream out; // deklaracja strumienia wyjscia do pliku
ifstream in;  // deklaracja strumienia wejscia z pliku

void generatorCiagow(int *Tab, int R, long int N) // funkcja realizujaca generowanie ciagow calkowitych liczb losowych z przedzialu od 0 do podanego przez uzytkownika N i zapisujaca je do pliku "losowe.txt"
{
    srand((unsigned)time(NULL)); // konfiguracja maszyny losujacej
    out.open("losowe.txt",ios::out); // otwarcie pliku "losowe.txt" do ktorego wypisane zostana wylosowane liczby
    for(int i=0; i<R; i++)
        {
            Tab[i] = rand()%(N +1); // wylosowanie do tablicy liczb z przedzialu od 0 do podanego przez uzytkownika N
        }
    for(int i=0; i<R-1; i++)
    {
        out<<Tab[i]<<" "; // wypisanie wylosowanych liczb do pliku "losowe.txt"
    }
    out<<Tab[R-1];
    out.close(); // zamkniecie strumienia wyjsciowego do pliku
};

void scalanie(int *Tab, int left, int middle, int right) // funkcja scalajaca podtablice
{
   int i, j, k; // zmienne iteracyjne do petli "for"
   int leftSize = middle-left+1; // rozmiar lewej podtablicy
   int rightSize = right-middle; // rozmiar prawej podtablicy
   int *leftTab = new int[leftSize]; // deklaracja lewej podtablicy o odpowiednim rozmiarze
   int *rightTab = new int[rightSize]; // deklaracja prawej podtablicy o odpowiednim rozmiarze
   for(i = 0; i<leftSize; i++)
    {
        leftTab[i] = Tab[left+i]; // wypelnienie lewej podtablicy odpowiednimi liczbami z tablicy
    };
   for(j = 0; j<rightSize; j++)
    {
        rightTab[j] = Tab[middle+1+j];  // wypelnienie prawej podtablicy odpowiednimi liczbami z tablicy
    };
   i = 0; j = 0; k = left; // przypisanie wartosci zmiennym iteracyjnym potrzebnym do petli "for"
   while(i < leftSize && j<rightSize)
    {
        if(leftTab[i] <= rightTab[j])
        {
            Tab[k] = leftTab[i];    // scalenie podtablic z tablica glowna
            i++;                    // inkrementacja zmiennej i
        }
        else
        {
            Tab[k] = rightTab[j];
            j++; // inkrementacja zmiennej j
        }
      k++; // inkrementacja zmiennej k
    };
   while(i<leftSize)
    {
        Tab[k] = leftTab[i];    // dodatkowy element z lewej podtablicy
        i++; k++;   // inkrementacja zmiennej k i zmiennej i
    }
   while(j<rightSize)
    {
        Tab[k] = rightTab[j];   //dodatkowy element z prawej podtablicy
        j++; k++; // inkrementacja zmiennej k i zmiennej j
    }
};

void mergeSort(int *Tab, int left, int right) // funkcja realizujaca sortowanie przez scalanie
```

```
78  {
79     int middle; // zmienna przechowujaca srodek tablicy
80     if(left < right)
81      {
82          middle = left+(right-left)/2; // wyznaczenie srodka tablicy, potrzebnego do podzialu na
    2 tablice,
83          mergeSort(Tab, left, middle); // sortowanie rekurencyjnie lewej czesci tablicy
84          mergeSort(Tab, middle+1, right);    // sortowanie rekurencyjnie prawej czesci tablicy
85          scalanie(Tab, left, middle, right); // scalanie lewej i prawej czesci tablicy w calosc
86      }
87  };
88
89  void quickSort(int *Tab, int left, int right) // funkcja realizujaca sortowanie quickSort
90  {
91      int x; // deklaracja zmiennej x
92      int y; // deklaracja zmiennej y
93      int pivot; // deklaracja zmiennej pivot, bedacej elementem podzialowym
94    x = (left + right)/2; // przypisanie wartosci do zmiennej x
95    pivot = Tab[x];   // przypisanie do zmiennej pivot wartosci zawartej w tablicy pod indeksem x
96    Tab[x] = Tab[right]; // przypisanie do komorki tablicy pod indeksem x, wartosci zawartej pod
    indeksem right
97      for(y=x=left; x < right; x++)
98          {
99          if(Tab[x] < pivot)
100             {
101                 swap(Tab[x], Tab[y]); // zamiana miejscami wartosci tablicy zawartych pod
    indeksami x i y
102                 y++; // inkrementacja zmiennej y
103             }
104         };
105   Tab[right] = Tab[y]; // przypisanie do komorki tablicy zawartej pod indeksem right, wartosci
    z komorki zawartej pod indeksem y
106   Tab[y] = pivot;   // przypisanie do komorki tablicy pod indeksem y, wartosci zawartej pod
    indeksem pivot
107     if(left < y - 1)
108         {
109             quickSort(Tab, left, y - 1); // rekurencyjne wywolanie algorytmu sortujacego
    "quickSort" dla lewej podtablicy
110         };
111     if(y + 1 < right)
112         {
113             quickSort(Tab, y + 1, right); // rekurencyjne wywolanie algorytmu sortujacego
    "quickSort" dla prawej podtablicy
114         };
115 }
116
117 int main()
118 {
119
120     cout<<"Witaj w programie!"<<endl; // przejrzyste menu programu, ulatwiajace korzystanie z
    niego
121     cout<<"Mozliwe dzialania: "<<endl;
122     cout<<"1. Wprowadzenie danych z klawiatury."<<endl<<"2. Wczytywanie danych z
    pliku."<<endl<<"3. Liczby pseudolosowe."<<endl<<"4. Generowanie ciagu do pliku losowe.txt"<<endl;
123     cout<<"Wybierz numer dzialania: ";
124     cin>>nr_dzialania; //
125     if (!cin)
126             {
127                 cout<<"Zle dokonales wyboru! :/"<<endl; // sprawdzenie czy wprowadzony znak
    jest wartoscia int, jesli nie wyswietlany jest komunikat
128                 cin.clear();
129                 cin.sync();
130             }
131     else
132     {
133     switch(nr_dzialania)    // funkcja "switch" zalezna od numeru dzialania ktory uzytkownik
    wybierze
134     {
135         case 1:
136         {
137             cout<<"Podaj rozmiar tablicy: ";
138             cin>>R; // wczytanie rozmiaru tablicy
139             if (!cin)
140             {
141                 cout<<"Podany rozmiar tablicy nie jest liczba. Program zostal
    zakonczony."<<endl; // sprawdzenie czy wprowadzony znak jest wartoscia int, jesli nie
    wyswietlany jest komunikat
142                 cin.clear();
143                 cin.sync();
144             }
145             else
146             {
147             int *Tablica = new int[R]; // deklaracja tablicy dynamicznej o rozmiarze podanym
    przez uzytkownika
```

```
148             int i=0;
149                     for(i=0;i<R;i++) // petla for potrzebna do wypeleniena naszej tablicy liczbami
150                     {
151                         cout<<"Podaj "<<i+1<<" element tablicy: ";
152                         cin>>Tablica[i]; // wczytanie podanego elementu do odpowiedniej komorki tablicy
153                         if(!cin)
154                         {
155                             cout<<"Wprowadz liczbe!"<<endl; // jesli zostanie podany znak inny niz liczba program wyswietli komunikat
156                             cin.clear ();
157                             cin.sync ();
158                             i--; // dekrementacja zmiennej i
159                     }
160                     else
161                         continue;
162                     }
163             cout<<endl;
164             cout<<"1. Sortowanie quickSort."<<endl<<"2. Sortowanie przez scalanie."<<endl;
165             cout<<"Wybierz algorytm sortowania: "; // menu wyboru algorytmu, ktorym chcemy posortowac nasze liczby
166             cin>>sposob;
167
168             switch(sposob)  // funkcja "switch" zalezna od sposobu sortowania ktory uzytkownik wybierze
169             {
170                 case 1:
171                 {
172                     cout<<"Sortuje tablice, algorytmem quicksort. Prosze czekac!"<<endl<<endl;
173                     start = clock(); // zapisanie liczby cykli procesora w momencie rozpoczecia algorytmu sortowania
174                     quickSort(Tablica,0,R-1); // sortowanie metoda quickSort
175                     stop = clock();     // zapisanie liczby cykli procesora w momencie zakonczenia algorytmu sortowania
176                     czas = (double)(stop-start)/CLOCKS_PER_SEC;  // podzielenie liczby cykli przez stala "CLOCKS_PER_SEC" i zrzutowanie na typ double, aby otrzymac czas w sekundach
177                     cout<<"Czas sortowania: "<<czas<<endl<<endl; // wyswietlenie czasu dzialania algorytmu sortowania
178                     out.open("wynik.txt",ios::out); // otwarcie pliku "wynik.txt" do ktorego zostanie zapisany wynik dzialania programu
179                     cout<<endl;
180                     cout<<"Posortowana tablica: ";
181                     for(int i=0;i<R;i++)
182                     {
183                         cout<<" | "<<Tablica[i];     // wyswietlenie posortowanej tablicy na ekranie
184                         out<<" | "<<Tablica[i];     // wypisanie posortowanej tablicy do pliku
185                     };
186                     cout<<" | ";
187                     out<<" | ";
188                     out.close();    // zamkniecie strumienia wyjsciowego do pliku
189                     cout<<endl<<endl;
190                 }break;
191                 case 2:
192                 {
193                     cout<<"Sortuje tablice, algorytmem przez scalanie. Prosze czekac!"<<endl<<endl;
194                     start = clock(); // zapisanie liczby cykli procesora w momencie rozpoczecia algorytmu sortowania
195                     mergeSort(Tablica,0,R-1); // sortowanie metoda sortowania przez scalanie (mergeSort)
196                     stop = clock(); // zapisanie liczby cykli procesora w momencie zakonczenia algorytmu sortowania
197                     czas = (double)(stop-start)/CLOCKS_PER_SEC; // podzielenie liczby cykli przez stala "CLOCKS_PER_SEC" i zrzutowanie na typ double, aby otrzymac czas w sekundach
198                     cout<<"Czas sortowania: "<<czas<<endl<<endl; // wyswietlenie czasu dzialania algorytmu sortowania
199                     out.open("wynik.txt",ios::out); // otwarcie pliku "wynik.txt" do ktorego zostanie zapisany wynik dzialania programu
200                     cout<<endl;
201                     cout<<"Posortowana tablica: ";
202                     for(int i=0;i<R;i++)
203                     {
204                         cout<<" | "<<Tablica[i]; // wyswietlenie posortowanej tablicy na ekranie
205                         out<<" "<<Tablica[i]; // wypisanie posortowanej tablicy do pliku
206                     };
207                     cout<<" | ";
208                     out<<" ";
209                     out.close(); // zamkniecie strumienia wyjsciowego do pliku
210                     cout<<endl<<endl;
211                 }
212             }
213             delete []Tablica;   // usuniecie tablicy dynamicznej z systemu (zwrocenie pamieci
```

```
zajetej przez tablice do systemu)
214             }
215         }break;
216         case 2:
217         {
218         string plik_wejsciowy;
219         cout<<"Podaj sciezke pliku: ";
220         cin>>plik_wejsciowy;
221         in.open(plik_wejsciowy, ios::in); // otwarcie pliku zawierajacego dane wejsciowe
222             if(in.good()==false) // jezeli wystapi jakis problem w czasie otwierania pliku
program wyswietli komunikat "Nie mozna otworzyc pliku"
223             {
224                 cout<<"Nie mozna otworzyc pliku"<<endl;
225                 exit(0);
226             }
227             else
228             {
229             int temp, ilosc = 0; // deklaracja licznika ilosci danych w pliku "ilosc" oraz
zmienniej tymczasowej "temp"
230                if(ifstream(plik_wejsciowy, ios::ate).tellg()) // jezeli plik nie bedzie
zawieral danych(bedzie pusty) program wyswietli komunikat "Plik jest pusty"
231                 {
232                 while(!in.eof())// petla wykonuje sie dopoki dane w pliku wejsciowym sie nie
skoncza
233                 {
234                     in>>temp;       // przypisanie danych zawartych w pliku do zmiennej 'a' w
celu policzenia ich ilosci
235                     ilosc++;  // wskaznik ilosci danych zawartych w pliku, potrzebny do
stworzenia tablicy
236                 }
237                 in.close();
238                 cout<<"Ilosc: "<<ilosc<<endl;
239
240                 int *T = new int[ilosc];    // deklaracja tablicy o wielkosci rownej ilosci
elementow w pliku tekstowym
241                 int m=0; // zmienna iteracyjna
242                 in.open(plik_wejsciowy, ios::in);
243                 while(!in.eof()) // petla wykonuje sie dopoki dane w pliku wejsciowym sie nie
skoncza
244                 {
245                     in >> T[m];      // wpisywanie liczb zawartych w pliku do tablicy "T"
246                     m++;    // inkrementacja zmiennej m
247                 }
248                 in.close(); // zamkniecie strumienia wejsciowego z pliku
249                 cout<<endl;
250                 /*for(int i=0;i<ilosc; i++)
251                 {
252                     cout<<" | "<<T[i]; // opcjonalne wypisywanie liczb wczytanych z pliku na
ekranie, wylaczone ze wzgledu na duza ilosc elementow
253                 };
254                 cout<<" | "<<endl;*/
255
256                 cout<<"1. Sortowanie quickSort."<<endl<<"2. Sortowanie przez scalanie."<<endl;
257                 cout<<"Wybierz algorytm sortowania: ";
258                 cin>>sposob;
259
260                 R = ilosc;
261                 switch(sposob)  // funkcja "switch" zalezna od sposobu sortowania ktory
uzytkownik wybierze
262                 {
263                     case 1:
264                     {
265                         cout<<"Sortuje tablice, algorytmem quicksort. Prosze
czekac!"<<endl<<endl;
266                         start = clock();    // zapisanie liczby cykli procesora w momencie
rozpoczecia algorytmu sortowania
267                         quickSort(T,0,R-1); // sortowanie metoda quickSort
268                         stop = clock(); // zapisanie liczby cykli procesora w momencie
zakonczenia algorytmu sortowania
269                         czas = (double)(stop-start)/CLOCKS_PER_SEC; // podzielenie liczby cykli
przez stala "CLOCKS_PER_SEC" i zrzutowanie na typ double, aby otrzymac czas w sekundach
270                         cout<<"Czas sortowania: "<<czas<<endl<<endl;    // wyswietlenie czasu
dzialania algorytmu sortowania
271                         out.open("wynik.txt",ios::out); // otwarcie pliku "wynik.txt" do
ktorego zostanie zapisany wynik dzialania programu
272                         cout<<endl<<endl;
273                         /*cout<<"Posortowana tablica: ";*/
274                         for(int i=0;i<R;i++)
275                         {
276                             /*cout<<" | "<<T[i];*/  // opcjonalne wyswietlenie posortowanej
tablicy na ekranie, wylaczone ze wzgledu na duza ilosc elementow
277                             out<<" | "<<T[i]; // wypisanie posortowanej tablicy do pliku
278                         };
279                         /*cout<<" | ";*/
```

```
280                         out<<" | ";
281                         out.close();    // zamkniecie strumienia wyjsciowego do pliku
282                         cout<<endl<<endl;
283                     }break;
284                     case 2:
285                     {
286                         cout<<"Sortuje tablice, algorytmem sortowania przez scalanie. Prosze
    czekac!"<<endl<<endl;
287                         start = clock();    // zapisanie liczby cykli procesora w momencie
    rozpoczecia algorytmu sortowania
288                         mergeSort(T,0,R-1); // sortowanie metoda sortowania przez scalanie
    (mergeSort)
289                         stop = clock(); // zapisanie liczby cykli procesora w momencie
    zakonczenia algorytmu sortowania
290                         czas = (double)(stop-start)/CLOCKS_PER_SEC; // podzielenie liczby cykli
    przez stala "CLOCKS_PER_SEC" i zrzutowanie na typ double, aby otrzymac czas w sekundach
291                         cout<<"Czas sortowania: "<<czas<<endl<<endl;    // wyswietlenie czasu
    dzialania algorytmu sortowania
292
293                         cout<<endl<<endl;
294                         /*cout<<"Posortowana tablica: ";*/
295                         out.open("wynik.txt",ios::out); // otwarcie pliku "wynik.txt" do
    ktorego zostanie zapisany wynik dzialania programu
296                         for(int i=0;i<R-1;i++)
297                         {
298                             /*cout<<" | "<<T[i];*/  // opcjonalne wyswietlenie posortowanej
    tablicy na ekranie, wylaczone ze wzgledu na duza ilosc elementow
299                             out<<T[i]<<" ";     // wypisanie posortowanej tablicy do pliku
300                         };
301                         /*cout<<T[R-1]<<" | ";*/
302                         out<<T[R-1];
303                         out.close();    // zamkniecie strumienia wyjsciowego do pliku
304                         cout<<endl<<endl;
305                     }
306                 }
307             delete [] T;    // usuniecie tablicy dynamicznej z systemu (zwrocenie pamieci
    zajetej przez tablice do systemu)
308             }
309             else
310             {
311                 cout<<"Plik jest pusty."<<endl;
312             };
313             }
314         }break;
315         case 3:
316         {
317             cout<<"Podaj rozmiar tablicy: ";
318             cin>>R;
319             if (!cin)
320             {
321                 cout<<"Podany rozmiar tablicy nie jest liczba. Program zostal
    zakonczony."<<endl; // sprawdzenie czy wprowadzony znak jest wartoscia int, jesli nie
    wyswietlany jest komunikat
322                 cin.clear();
323                 cin.sync();
324             }
325             else
326             {
327             int *Tablica = new int[R];  // deklaracja tablicy o wielkosci podanej przez
    uzytkownika
328             cout<<"Zakres losowania [ 0, N ], podaj N: ";
329             cin>>N;
330             generatorCiagow(Tablica,R,N); // wywolanie funkcji generujacej liczby calkowite z
    przedzialu od 0 do podanego N i zapisujacej je do pliku "losowe.txt"
331             cout<<endl;
332             for(int i=0;i<R;i++)
333             {
334                 cout<<" | "<<Tablica[i];// wypisanie wylosowanych liczb na ekranie
335             };
336             cout<<" | "<<endl<<endl;
337             in.open("losowe.txt", ios::in); // otwarcie pliku "losowe.txt" zawierajacego
    elementy wylosowane przez generator ciagow
338             if(in.good()==false) // jezeli wystapi jakis problem w czasie otwierania pliku
    program wyswietli komunikat "Nie mozna otworzyc pliku"
339             {
340                 cout<<"Nie mozna otworzyc pliku"<<endl;
341                 exit(0);
342             }
343             else
344             {
345             int j=0; // zmienna iteracyjna
346             int *Tab = new int[R];  // deklaracja tablicy o wielkosci podanej przez uzytkownika
347             while(!in.eof())// petla wykonuje sie dopoki dane w pliku wejsciowym sie nie skoncza
348                 {
```

```
                    in>>Tab[j]; // wczytanie liczb z pliku do tablicy
                    j++;    // inkrementacja zmiennej j
                }
            in.close();
            cout<<"1. Sortowanie quickSort."<<endl<<"2. Sortowanie przez scalanie."<<endl;
            cout<<"Wybierz algorytm sortowania: ";
            cin>>sposob;

            switch(sposob)  // funkcja "switch" zalezna od sposobu sortowania ktory uzytkownik wybierze
            {
                case 1:
                {
                    cout<<"Sortuje tablice, algorytmem quicksort. Prosze czekac!"<<endl<<endl;
                    start = clock();    // zapisanie liczby cykli procesora w momencie rozpoczecia algorytmu sortowania
                    quickSort(Tab,0,R-1);   // sortowanie metoda quickSort
                    stop = clock(); // zapisanie liczby cykli procesora w momencie zakonczenia algorytmu sortowania
                    czas = (double)(stop-start)/CLOCKS_PER_SEC; // podzielenie liczby cykli przez stala "CLOCKS_PER_SEC" i zrzutowanie na typ double, aby otrzymac czas w sekundach
                    cout<<"Czas sortowania: "<<czas<<endl<<endl;    // wyswietlenie czasu dzialania algorytmu sortowania
                    out.open("wynik.txt",ios::out); // otwarcie pliku "wynik.txt" do ktorego zostanie zapisany wynik dzialania programu
                    cout<<endl<<endl;
                    cout<<"Posortowana tablica: ";
                    for(int i=0;i<R;i++)
                    {
                        cout<<" | "<<Tab[i];    // wyswietlenie posortowanej tablicy na ekranie
                        out<<" | "<<Tab[i];     // wypisanie posortowanej tablicy do pliku
                    };
                    cout<<" | ";
                    out<<" | ";
                    out.close();    // zamkniecie strumienia wyjsciowego do pliku
                    cout<<endl<<endl;
                }break;
                case 2:
                {
                    cout<<"Sortuje tablice, algorytmem przez scalanie. Prosze czekac!"<<endl<<endl;
                    start = clock();    // zapisanie liczby cykli procesora w momencie rozpoczecia algorytmu sortowania
                    mergeSort(Tab,0,R-1);   // sortowanie metoda sortowania przez scalanie (mergeSort)
                    stop = clock(); // zapisanie liczby cykli procesora w momencie zakonczenia algorytmu sortowania
                    czas = (double)(stop-start)/CLOCKS_PER_SEC; // podzielenie liczby cykli przez stala "CLOCKS_PER_SEC" i zrzutowanie na typ double, aby otrzymac czas w sekundach
                    cout<<"Czas sortowania: "<<czas<<endl<<endl; // wyswietlenie czasu dzialania algorytmu sortowania
                    out.open("wynik.txt",ios::out); // otwarcie pliku "wynik.txt" do ktorego zostanie zapisany wynik dzialania programu
                    cout<<endl<<endl;
                    cout<<"Posortowana tablica: ";
                    for(int i=0;i<R;i++)
                    {
                        cout<<" | "<<Tab[i];    // wyswietlenie posortowanej tablicy na ekranie
                        out<<" "<<Tab[i];     // wypisanie posortowanej tablicy do pliku
                    };
                    cout<<" | ";
                    out<<" ";
                    out.close();    // zamkniecie strumienia wyjsciowego do pliku
                    cout<<endl<<endl;
                }break;
            }delete Tab;    // usuniecie tablicy dynamicznej z systemu (zwrocenie pamieci zajetej przez tablice do systemu)
            }
            delete Tablica; // usuniecie tablicy dynamicznej z systemu (zwrocenie pamieci zajetej przez tablice do systemu)
            }
        }break;
        case 4:
        {
          cout<<"Podaj rozmiar tablicy: ";
            cin>>R;
            if (!cin)
            {
                cout<<"Podany rozmiar tablicy nie jest liczba. Program zostal zakonczony."<<endl; // sprawdzenie czy wprowadzony znak jest wartoscia int, jesli nie wyswietlany jest komunikat
                cin.clear();
                cin.sync();
            }
```

```
416             else
417             {
418             int *Tablica = new int[R]; // deklaracja tablicy o wielkosci podanej przez
    uzytkownika, do ktorej wylosowane zostana liczby
419             cout<<"Zakres losowania [ 0, N ], podaj N: ";
420             cin>>N;
421             generatorCiagow(Tablica,R,N); // wywolanie funkcji generujacej liczby calkowite z
    przedzialu od 0 do podanego N i zapisujacej je do pliku "losowe.txt"
422             cout<<endl;
423             cout<<"Gotowe!"<<endl;
424             delete [] Tablica; // usuniecie tablicy dynamicznej z systemu (zwrocenie pamieci
    zajetej przez tablice do systemu)
425             }
426         }
427     };
428     };
429     getch();
430
431     return 0;
432 }
433
```