Zdrojový kód riešenia zadania príkladu 8.1 rozšírte tak, aby program po zadaní počtu vygenerovaných komplexných čísiel *n < 11* vypísal obsahy polí *ccisla1ptp* a *ccisla2ptp* s vygenerovanými komplexnými číslami vrátane hodnôt ich modulov na konzolu a taktiež aby program vypísal obsahy týchto polí, komplexné čísla s ich modulmi, po ich usporiadaní funkciami *QuickSort* a *qsort*.

Pre vypisovanie polí si vytvorte funkciu s nasledovnou deklaráciou

void vypis(CPLX \*data[], int n); // *funkcia vypise 'n' prvkov pola 'data'*

, ktorú môžete potom podľa potreby vo funkcii *main* opakovane volať, avšak vždy s relevantnými parametrami.

#include <math.h>

#include <time.h>

#define MAX\_CISLO\_1 9

#define MAX\_CISLO\_2 18

#include <iostream>

using namespace std;

struct CPLX { double re, im; };

/\* Funkcia ocakava v 1 formalnom argumente ukazovatel na na strukturovu premennu typu 'CPLX'.

Funkcia vygeneruje komplexne cislo pomocou generatora pseudonahodnych cisiel 'rand'.

\*/

void generujCPLX(CPLX \*c)

{

c->re = 1 + (rand() % MAX\_CISLO\_1);

c->im = 1 + (rand() % MAX\_CISLO\_2);

}

/\* Funkcia ocakava v 1. formalnom argumente ukazovatel na strukturovu premennu typu 'CPLX'.

Funkcia vypocita modul komplexneho cisla (cize jeho velkost).

\*/

double abs(CPLX \*x)

{

return sqrt((x->re \* x->re) + (x->im \* x->im));

}

/\* Funkcia ocakava v 1. a 2. formalnom argumente ukazovatele na strukturove premenne typu 'CPLX'.

Funkcia podla velkosti vzajomne porovna strukturove premenne typu 'CPLX', na ktore ukazuju

ukazovatele 'a' a 'b'.

\*/

int porovnaj(CPLX \*a, CPLX \*b)

{

if (abs(a) > abs(b))

return 1;

else

return 0;

}

/\* Funkcia triedi pole strukturovych premennych algoritmom Quick sort.

Je jedno, ci je jej prvym argumentom 'CPLX \*\*data' alebo 'CPLX \*data[]', pretoze oba syntakticky

v nasom pripade predstavuju ukazovatel na pole ukazovatelov na strukturove premenne typu 'CPLX'.

Funkcia s oboma argumentmi funguje spravne.

\*/

void QuickSort(CPLX \*\*data, int lavy, int pravy)

{

if (lavy < pravy)

{

int i = lavy, j = pravy;

CPLX \*p = data[(lavy + pravy) / 2];

do

{

while (porovnaj(p, data[i]) > 0) i++;

while (porovnaj(data[j], p) > 0) j--;

if (i <= j)

{

CPLX \*tmp = data[i];

data[i] = data[j];

data[j] = tmp;

i++; j--;

}

} while (i <= j);

QuickSort(data, lavy, j);

QuickSort(data, i, pravy);

}

}

/\* Funkcia ocakava v 1 formalnom argumente ukazovatel na pole ukazovatelov na strukturove premenne typu 'CPLX'. V tele potom pracuje s prvkami 'data[i]' (cize s ukazovatelmi) tohto pola ukazovatelov na strukturove premenne typu 'CPLX'.

Funkcia skontroluje spravne usporiadanie prvkov pola, na ktore ukazuju ukazovatele 'data[i]',

cize usporiadanie strukturovych premennych typu 'CPLX' vzostupne podla ich velkosti.

Je jedno ci je jej prvym argumentom 'CPLX \*\*data' alebo 'CPLX \*data[]', pretoze oba syntakticky

v nasom pripade predstavuju ukazovatel na pole ukazovatelov na strukturove premenne typu 'CPLX'.

Funkcia s oboma argumentmi funguje spravne.

\*/

int skontroluj(CPLX \*data[], int n)

{

for (int i = 1; i < n; i++)

if (porovnaj(data[i - 1], data[i]) > 0)

return 0;

return 1;

}

/\* Funkcia ocakava v 1 a 2 formalnom argumente ukazovatele na polia ukazovatelov na strukturove premenne typu 'CPLX'. V tele potom pracuje s prvkami 'data1[i]' a 'data2[i]'(cize s ukazovatelmi) tychto poli ukazovatelov na strukturove premenne typu 'CPLX'.

Funkcia skontroluje zhodnost prvkov poli strukturovych premennych typu 'CPLX', na ktore ukazuju ukazovatele 'data1[i]' a 'data2[i]'.

Je jedno, ci je jej 1. a 2. argumentom 'CPLX \*\*data1' a 'CPLX \*\*data2' alebo 'CPLX \*data1[]'

a 'CPLX \*data2[]', pretoze oboje syntakticky v nasom pripade predstavuju ukazovatele na polia

ukazovatelov na strukturove premenne typu 'CPLX'. Funkcia s oboma argumentmi funguje spravne.

\*/

int skontroluj(CPLX \*data1[], CPLX \*data2[], int n)

{

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

if (porovnaj(data1[i], data2[i]) != 0)

return 0;

return 1;

}

//porovnavacia funkcia do kniznicnej funkcie 'qsort'

int porovnajQptp(const void \*a, const void \*b)

{

if (abs(\*(CPLX \*\*)a) > abs(\*(CPLX \*\*)b))

return 1;

if (abs(\*(CPLX \*\*)a) < abs(\*(CPLX \*\*)b))

return -1;

return 0;

}

void vypis(CPLX \*data[], int n)

{

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

{

cout << data[i]->re << "+ " << data[i]->im << "i" << "(" << abs(data[i]) << ")" << endl;

}

}

int main()

{

int n, i;

cout << "vlozte pocet komplexnych cisiel, ktore ma program vygenerovat: ";

cin >> n;

//'ccisla1ptp' je ukazovatel na ukazovatel na CPLX

CPLX \*\*ccisla1ptp = new CPLX\*[n]; //alokovanie miesta pre n-prvkove POLE UKAZOVATELOV na

//CPLX, na ktore ukazuje ukazovatel 'ccisla1ptp'

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

ccisla1ptp[i] = new CPLX; //inicializacia ukazovatela 'ccisla1ptp[i]', vkladame do neho ukazovatel na

//pamatove miesto pre premennu typu CPLX

CPLX \*\*ccisla2ptp = new CPLX\*[n];

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

ccisla2ptp[i] = new CPLX;

/\* nastavenie startovacieho cisla generatora pseudonahodnych cisiel 'rand' tak, aby toto cislo bolo

vzdy ine, po novom zavolani funkcie 'main'. Takto tento generator vygeneruje vzdy ine

pseudonahodne cisla \*/

srand((unsigned)time(NULL));

/\* plnenie obsahu prvkov pola 'ccisla1ptp' vygenerovanymi komplexnymi cislami

a kopirovanie prvkov pola 'ccisla1ptp' do prvkov pola 'ccisla2ptp' \*/

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

{

generujCPLX(ccisla1ptp[i]);

ccisla2ptp[i]->re = ccisla1ptp[i]->re;

ccisla2ptp[i]->im = ccisla1ptp[i]->im;

// ccisla2ptp[i]=ccisla1ptp[i];

}

cout << "prvky pola ccisla1ptp pred ich usporiadanim funkciami Quicksort a Qsort " << endl;

vypis(ccisla1ptp, n);

cout << "prvky pola ccisla2ptp pred ich usporiadanim funkciami Quicksort a Qsort " << endl;

vypis(ccisla2ptp, n);

int k;

clock\_t c1, c2;

c1 = clock();

QuickSort(ccisla1ptp, 0, n - 1);

c2 = clock();

k = skontroluj(ccisla1ptp, n);

cout << "\nkontrola spravneho usporiadania pola 'ccisla1ptp' funkciou 'QuickSort': " << k << endl;

cout << " (trvanie usporiadania pola 'ccisla1ptp' funkciou 'QuickSort': " << (c2 - c1) /

CLOCKS\_PER\_SEC << " s)\n";

c1 = clock();

qsort((void \*)ccisla2ptp, n, sizeof(ccisla2ptp[0]), porovnajQptp);

c2 = clock();

k = skontroluj(ccisla2ptp, n);

cout << "\nkontrola spravneho usporiadania pola 'ccisla2ptp' funkciou 'qsort': " << k << endl;

cout << " (trvanie usporiadania pola 'ccisla2ptp' funkciou 'qsort': " << (c2 - c1) /

CLOCKS\_PER\_SEC << " s)\n";

k = skontroluj(ccisla1ptp, ccisla2ptp, n);

cout << "\nkontrola zhodnosti prvkov usporiadanych poli 'ccisla1ptp' a 'ccisla2ptp': " << k << endl;

cout << "prvky pola ccisla1ptp po ich usporiadanim funkciami Quicksort a Qsort " << endl;

vypis(ccisla1ptp, n);

cout << "prvky pola ccisla2ptp po ich usporiadanim funkciami Quicksort a Qsort " << endl;

vypis(ccisla2ptp, n);

delete ccisla1ptp[n - 1]; //mazeme pole ukazovatelov

delete[] ccisla1ptp; //mazeme ukazovatel na toto pole

delete ccisla2ptp[n - 2];

delete[] ccisla2ptp;

return 0;

}

vlozte pocet komplexnych cisiel, ktore ma program vygenerovat: 10

neusporiadane pole komplexnych cisiel 'ccisla1ptp':

3 + 15i (15.2971)

8 + 17i (18.7883)

4 + 8i (8.94427)

3 + 2i (3.60555)

1 + 18i (18.0278)

6 + 10i (11.6619)

6 + 13i (14.3178)

4 + 1i (4.12311)

6 + 12i (13.4164)

4 + 10i (10.7703)

neusporiadane pole komplexnych cisiel 'ccisla2ptp':

3 + 15i (15.2971)

8 + 17i (18.7883)

4 + 8i (8.94427)

3 + 2i (3.60555)

1 + 18i (18.0278)

6 + 10i (11.6619)

6 + 13i (14.3178)

4 + 1i (4.12311)

6 + 12i (13.4164)

4 + 10i (10.7703)

kontrola spravneho usporiadania pola 'ccisla1ptp' funkciou 'QuickSort': 1

(trvanie usporiadania pola 'ccisla1ptp' funkciou 'QuickSort': 0 s)

kontrola spravneho usporiadania pola 'ccisla2ptp' funkciou 'qsort': 1

(trvanie usporiadania pola 'ccisla2ptp' funkciou 'qsort': 0 s)

kontrola zhodnosti prvkov usporiadanych poli 'ccisla1ptp' a 'ccisla2ptp': 1

USPORIADANE pole komplexnych cisiel 'ccisla1ptp' funkciou 'QuickSort':

3 + 2i (3.60555)

4 + 1i (4.12311)

4 + 8i (8.94427)

4 + 10i (10.7703)

6 + 10i (11.6619)

6 + 12i (13.4164)

6 + 13i (14.3178)

3 + 15i (15.2971)

1 + 18i (18.0278)

8 + 17i (18.7883)

USPORIADANE pole komplexnych cisiel 'ccisla2ptp' funkciou 'qsort':

3 + 2i (3.60555)

4 + 1i (4.12311)

4 + 8i (8.94427)

4 + 10i (10.7703)

6 + 10i (11.6619)

6 + 12i (13.4164)

6 + 13i (14.3178)

3 + 15i (15.2971)

1 + 18i (18.0278)

8 + 17i (18.7883)