Apraksts

Šajā laboratorijas darbā tika pētīts *bubble sort* algoritms skaitļa rindas kārtošanai. Konkrēta parauga ietvaros algoritms tika pētīts ar 6 cipariem.

Algoritms un kods

Bubble sort ir kārtošanas algoritms, kurā, ja konstatē, ka cipars ir lielāks vai mazāks (atkarīgs no tā, kā grib kārtot skaitļus) blakusesošajam ciparam, tad šie skaitļi tiek mainīti ar vietām. Algoritmā ir 2 cikli: viens cikls iziet cauri visiem elementiem, bet otrs pārskata arī visus elementus, neskaitot tos, kuri jau pie pirmā cikla ir sakārtoti. Tā cikli turpinās līdz tiek iziets cauri visiem elementiem.

Vispirms tiek prasīts lietotājam ievadīt skaitļus:

```
int main(){
        int skaitlis;
        int rinda[6];
        for(skaitlis=0;skaitlis<6;skaitlis++){
                printf("Ludzu ievadiet %d.skaitli:\n",skaitlis);
                scanf("%d",&rinda[skaitlis]);
        }
```

Ilustrācija 1. Konkrētajā gadījumā tiks prasītas 6 reizes ievadīt skaitli, jo dotajai matricai ir 6 elementi. Taču kods arī strādās, ja pielāgos citu skaitu elementu. Jautājumu skaitam jābūt mazākam par matricas elementu skaitu, citādi būs kļūda.

Tālāk seko pats *bubble sort* algoritms:

```
void kartosana(int rinda[], int n){
        int i, j;
        for(i=0;i<n-1;i++){
                for(j=0;j<n-i-1;j++){
                        if(rinda[j]>rinda[j+1]){
                                int temp = rinda[j];
                                rinda[j] = rinda[j+1];
                                rinda[j+1] = temp;
                        }
                }
        }
}
```

Ilustrācija 2. Kā var redzēt, tiek izmantoti 2 cikli: viens liek darboties algoritmam uz visiem cipariem, otrs kārto ciparus, kuri nav sakārtoti.

Gadījumos, kad nepieciešams mainīt vietām locekļus, tiek izmantots *temporary* mainīgais, t.i., mainīgais, uz kura sākumā pasludinam, ka tas ir vienāds ar vienu elementu, tad mēs pasludinam, ka šis elements ir vienāds ar otru elementu, no kā seko, ka otrais elements ir vienāds ar *temp* mainīgajā esošo pirmo elementu. Tā norisinās elementu vērtību apmaiņa.

Tad, kad algoritms ir pabeigts, tad nepieciešams visu izvadīt:

```
void printArray(int rinda[], int size){
        int i;
        for(i=0;i<size;i++){
                printf("%d\n", rinda[i]);
        }
}
```

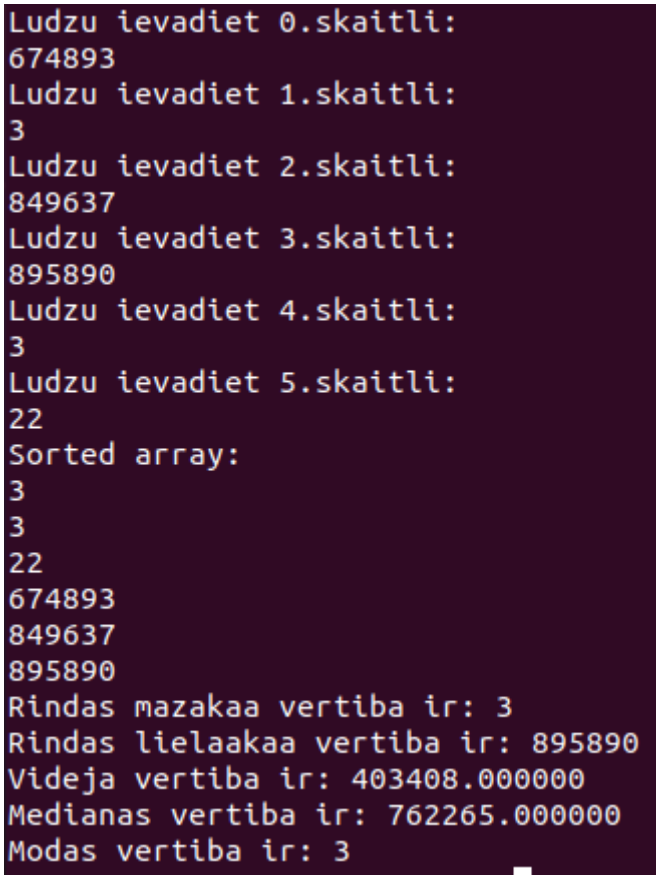
Ilustrācija 3. Manuāli rakstītā funkcija sakārtotās matricas elementu izvadīšanai.

Taču šajā funkcijā tiks izvadīti tikai sakārtotās matricas elementi. Atsevišķi ir nepieciešams izvadīt īpašos datus:

```
float videjais = (rinda[0]+rinda[1]+rinda[2]+rinda[3]+rinda[4]+rinda[5>
float mediana = (rinda[3]+rinda[4])/2;
int moda;
for(i=0;i<6;i++){
        for(j=0;j<6-i;j++){
                if(rinda[j]==rinda[j+1]){
                        moda = rinda[j];
                }
        }
}
printf("Rindas mazakaa vertiba ir: %d\n",rinda[0]);
printf("Rindas lielaakaa vertiba ir: %d\n",rinda[5]);
printf("Videja vertiba ir: %f\n",videjais);
printf("Medianas vertiba ir: %f\n",mediana);
printf("Modas vertiba ir: %d\n",moda);
```

Ilustrācija 4. Mazākā vērtība būs rindas 1.lockelis, lielākā vērtība - pēdējā(6.) vērtība, vidējais aritmētiskais, mediānas vērtība (vidējais starp 3. un 4.vērtību) un moda.

Par cik rinda šajā algoritmā tiek kārtota augošajā secībā, tad mazākā vērtība būs automātiski rindas 1.loceklis, savukārt, lielākā vērtība ir rindas pēdējais, t.i., 6. loceklis. Vidējā vērtība ir vidējais aritmētiskais rindas locekļiem un mediāna ir rindas vidējais loceklis (par cik ir pāra skaits locekļu, tad tas ir vidējais aritmētiskais 3. un 4. lockelim). Modas vērtība tiek noteikta sekojoši: tiek iziet cikls uz visiem lockeļiem un seko vēl viens cikls, lai salīdzinātu citus locekļus ar pārējiem. Ja kāds loceklis sakrīt ar citu locekli, tad modas vērtība kļūst par šo lockeli.



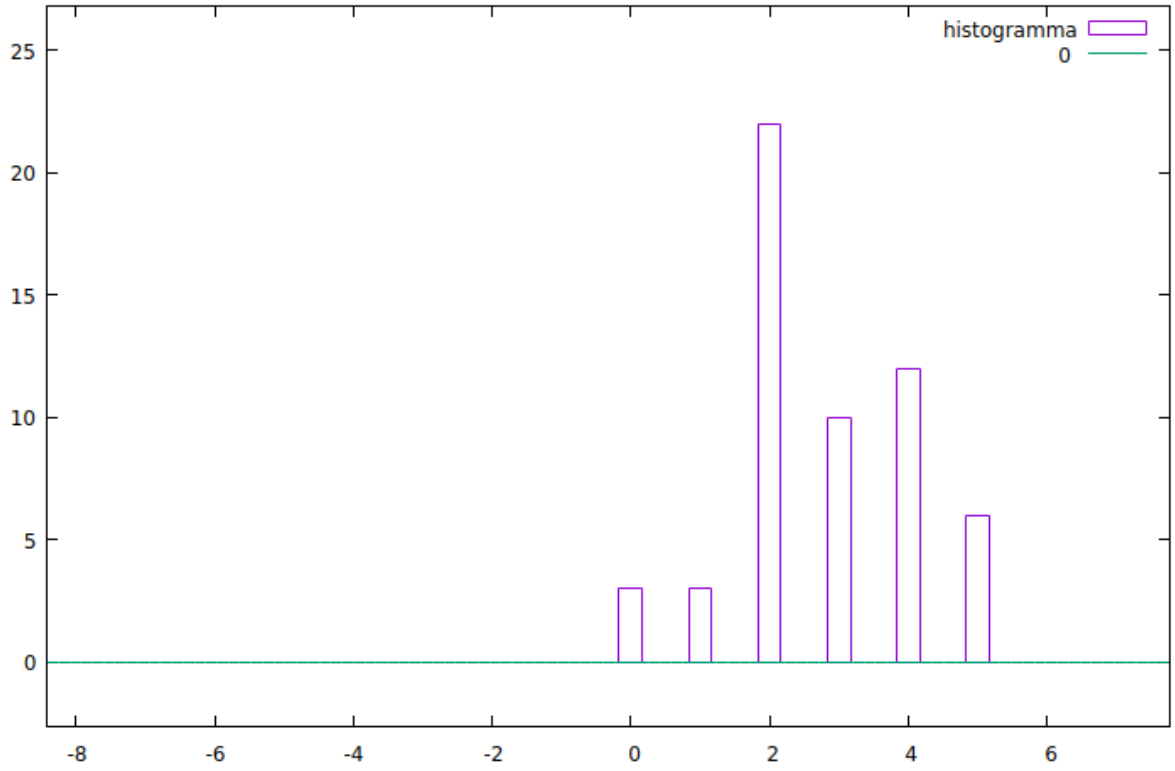
Ilustrācija 5. Ievadītie un pēc algoritma izvadītie dati.

Secinājumi

Kods strādā ļoti labi, par cik tiek prasīts lietotājam ievadīt datus, kā arī notiek *bubble sort* algoritma izpilde ar papildus informācijas izvadīšanu. Trūkumi sastādītajā kodā ir tas, ka šajā gadījumā ir fiksēts skaits rindas locekļu, tāpēc ir nepieciešams papildināt kodu, lai kods varētu darboties pie nenoteikta skaita rindas locekļu, kā arī modas vērtību var noteikt tikai vienu, tāpēc ir nepieciešams arī šeit papildināt kodu, lai varētu noteikt vairākas modas vērtības, ja tādas ir.

Bubble sort ir labs algoritms skaitļu kārtošanai, taču atsevišķā pētījumā ir nepieciešams noskaidrot, kāds ir tā kārtošanas ātrums, salīdzinot ar citiem algoritmiem, jo ir metodes, kuras var ātrāk sakārtot skaitļu rindas.

Pievienotie materiāli



Ilustrācija 6. Histogramma vienam mēģinājumam.