Dokumentácia

Lukáš Lechovič 07

Zadanie 5

Riešte problém, ktorý v štvorcovej matici zadaného ľubovoľného rozmeru N x N nájde riadok (riadky číslujte vo výstupe), ktorého absolútna hodnota súčtu jeho prvkov je maximum alebo minimum v danej matici. Všetky údaje vypíšte.

Analýza

Najprv si dajme trochu teórie z matematiky a to tým čo je to vlastne matica ? Matica je schematické usporiadanie prvkov do obdĺžnika alebo štvorca pozostávajúceho z riadkov a stĺpcov. Prídavne meno štvorcová znamená, že riadky a stĺpce si musia byť vždy rovné resp. matica musí byť v tvare štvorca. „nájde riadok, ktorého absolútna hodnota súčtu jeho prvkov je maximum alebo minimum v danej matici“ v skratke povedané tieto prvky z každého riadku spočíta a spraví z neho súčet. Súčet musí byť v absolútnej hodnote. To spravíme tak, že pokiaľ bude náš súčet menší ako 0 tak sa vynásobí -1. Naše súčty si program následne uloží a bude ich porovnávať s najmenšou a najväčšou hodnotou.

Implementácia pre používateľa

Program je veľmi jednoduchý a od používateľa žiada iba dva parametre. Veľkosť matice a rozhodnutie od používateľa či si chce vyplniť maticu sám alebo si ju nechá náhodne vygenerovať. Program priamo komunikuje s používateľom a nechce od používateľa nič zložité, preto si myslím že s programom by nemali mať problém ani starší ľudia a ľudia so základnými znalosťami s PC.

Po zadaní korektných hodnôt sa používateľovi zobrazí matica kde vždy na začiatku nového riadku matice je číslo riadku danej matice. Matica je čitateľná, priehľadná bez žiadnych rušivých elementov, pre následne lepšiu orientáciu a prácu s maticou. Pod maticou tu máme výpisy absolútnej hodnoty súčtov, kde je jasne napísané číslo riadku, aký ma súčet, a porovnanie s najmenšou a najväčšou hodnotou.

Program sa na koniec spýta používateľa či si nechce program spustiť znova.

Implementácia pre programátora

Pri prvotnom spustení programu používateľa oboznámi z riešenou problematikou a následne čaká program na zadanie príslušnej veľkosti matice. Input ako taký je ošetrený funkciou aby používateľ mohol písať iba čísla, plus je ošetrený pomocou cyklu do while s podmienkou na konci, ktorá slúži na to aby používať mohol zadať iba číslo väčšie ako 1. Po zadaní korektnej hodnoty si používateľ môže vybrať či si chce maticu vyplniť sám alebo si ju nechá vygenerovať. Tento input funguje na podobnom princípe ako input spomínaný vyššie iba z podmienkou že môže zadať iba 1 – teda vyplniť si maticu sám alebo 2 - vygenerovať si ju. V zloženej podmienke je teda dané, že číslo musí byť menšie ako 2 a zároveň väčšie ako 0. Program pri zistený zlých hodnôt nespadne a opätovne čaká na znovu ich zadanie.

Následne po vybratí, na pozadí funguje switch ktorí tvorí 2 case. Pokiaľ si používateľ vyberie vyplniť maticu sám alebo si ju nechá vygenerovať princíp týchto 2 casov je skoro rovnaký, mení sa iba spôsob zápisu prvkov do matice resp. výberu prispôsobenej funkcii na to. Pod switchom sa už iba nachádza for cyklus na výpočet súčtov z matice. Hotové súčty aj z absolútnou hodnotou sa ukladajú do poľa aby ich bolo možné znova porovnávať. Následne sa program spýta či si chce používateľ program spustiť znova. Výstup je ošetrený ako výstupy vyššie. Ak áno program sa celý zopakuje, ak nie program skončí.

Main()

1. Deklarácia premenných

```
int n, vyber = 0, i, j;
int pom = 0,znova = 0;
double sum, min, max;
```

2. Zadávanie veľkosti matice

```
do
{
    printf("\nZadajte velkost stvorcovej matice: ");
    n = osetrenie_vstupu(pom);
}
while (n < 2);
```

3. Zadávanie výpisu matice

```
do
{
    printf("\n\n(1) Prvky si zadate sami\n(2) Prvky si nechate nahodne vygenerovat\nZadajte 1 alebo 2: ");
    vyber = osetrenie_vstupu(pom);
}
while (vyber <= 0 || vyber > 2);
```

4. Výber matice

```
switch (vyber)
{
case 1:
    zadavanie_hodnot(n, arr);
    break;
case 2:
    generovanie_hodnot(n, arr);
    break;
}
```

5. Výpis extrémov

```
min = hladanie_minima(n, arr);
max = hladanie_maxima(n, arr);

printf("Najmensia hodnota v matici je %0.02lf\n", min);
printf("Najvacsia hodnota v matici je %0.02lf\n", max);
```

6. Súčet jednotlivých riadkov, porovnanie a výpis

```
printf("\n\tSUCET\t\t\tMIN\t\t\tMAX");

for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        sum += arr[i][j];
    }
    sum = absolutna_hodnota(sum);
    printf("\n(%d)\t%0.02lf\t", i + 1, sum);
    (sum != min) ? (sum > min ? printf("\t>\t%0.02lf\t", min) : printf("\t<\t%0.02lf\t", min)) : printf("\t=\t%0.02lf\t", min);
    (sum != max) ? (sum > max ? printf("\t>\t%0.02lf\t", max) : printf("\t<\t%0.02lf\t", max)) : printf("\t=\t%0.02lf\t", max);
    sucet[i] = sum;
    sum = 0;
}
```

Funkcie

- Deklarácia funkcií

```
int osetrenie_vstupu(int pom);
void uvolnenie(double **array, int n);
double **alokacia(int n);
void zadavanie_hodnot(int n, double **array);
void generovanie_hodnot(int n, double **array);
double nahodnecisla(double mini, double maxi);
double hladanie_minima(int n, double **array);
double hladanie_maxima(int n, double **array);
double absolutna_hodnota(double cislo);
```

- Ošetrenie vstupu

Tento algoritmus nie je môj, našiel som ho na internete. Algoritmus som si prerobil do funkcie a aby bol funkčný a použiteľný aj viac krát. Funkcia *osetrenie_vstupu()* je použitá v prvých 2 inputoch a na konci kde sa používateľa pýta či si chce program spustiť znova. Používateľ musí zadať iba kladné celé číslo. Písmená, symboly zadávať nejdú a ani sa vo výstupe neukazujú. Funkcia funguje na cykle *while(1),* kde vstup funguje cez *getch().* Podmienka If mi pomáha aby input bral do úvahy a vypísal čísla z ASCII tabuľky od 0 – 9. Číslo 13 v ASCII tabuľke znamená enter, takže po jeho stlačený cyklus skončí a číslo vráti.
V prípade že sa používateľ pomýli a zadá iné číslo iné ako chcel tak si program musí spustiť odznova.

```
int osetrenie_vstupu(int pom)
{
    int cislo = 0, ch;
    while (1)
    {
        ch = getch();
        if (ch >= 48 && ch <= 57)
        {
            printf("%c", ch);
            cislo = cislo * 10 + (ch - 48);
        }

        if (ch == 13)
            break;
    }
    return cislo;
}
```

- Uvoľnenie

Funkcia *uvolnenie()* je použitá na konci programu v maine, a vo funkcií *alokacia().* Zaisťuje že sa matica po skončení alebo v prípade nedostatku pamäte vyčistí a vymaže.

```c
void uvolnenie(double **array, int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        free(array[i]);
    }
    free(array);
    array = NULL;
}
```

- Alokácia

Funkcia ***alokacia()* slúži na alokáciu pamäte danej matice. V prípade nedostatku pamäte program vypíše chybovú hlášku a skončí.

```c
double **alokacia(int n)
{
    double **matica;
    int i;
    if ((matica = (double **)malloc(n * sizeof(double *))) == NULL)
    {
        printf("Nedostatok pamate. Program konci.");
        exit(1);
    }
    for (i = 0; i < n; i++)
    {
        if ((matica[i] = (double *)malloc(n * sizeof(double))) == NULL)
        {
            printf("Nedostatok pamate. Program konci.");
            uvolnenie(matica, i);
            exit(1);
        }
    }
    return matica;
}
```

- Zadávanie hodnôt

Funkcia *zadavanie_hodnot()* sa nachádza v prvom 1. case a je pre manuálne vpisovanie čísiel do matice. Vo funkcií funguje vnorený for cyklus, opakuje sa podľa toho aká veľká je matica. Formátovanie textu je priehľadné a používateľ by nemal mat problém s tým ktorý prvok momentálne vypisuje do matice. Používateľ môže zadávať reálne čísla.

```c
void zadavanie_hodnot(int n, double **array)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        printf("\n%d. riadok\n", i + 1);
        for (j = 0; j < n; j++)
        {
            printf("Prvok [%d][%d]: ", i + 1, j + 1);
            scanf("%lf", &array[i][j]);
        }
    }
}
```

- Generovanie hodnôt

Funkcia *generovanie_hodnot()* sa nachádza v 2. case a slúži aj pre výpis a vygenerovanie matice s náhodnými reálnymi číslami ktoré fungujú cez ďalšiu funkciu, v tejto sa iba volá a zapisuje a hodnoty sa ukladajú. *Double mini* a *maxi* sú pre výber rozsahu čísel.

```
void generovanie_hodnot(int n, double **array)
{
    int i, j;
    double mini = -100, maxi = 100;
    srand(time(NULL));
    for (i = 0; i < n; i++)
    {
        printf("(%d)", i + 1);
        for (j = 0; j < n; j++)
        {
            array[i][j] = nahodnecisla(mini, maxi);
            printf("\t%0.02lf\t", array[i][j]);
        }
        printf("\n");
    }
}
```

- Náhodne čísla

Funkcia *nahodnecisla()* je mnou osvojená ale nie moja, patrí dobrému spolužiakovi na vygenerovanie náhodných reálnych čísel. Funkcia je použitá vo funkcií *generovanie_hodnot().*

```
double nahodnecisla(double mini, double maxi)
{
    double nahodne = ((double)rand()) / RAND_MAX;
    double rozsah = (maxi - mini) * nahodne;
    double cisla = mini + rozsah;
    return cisla;
}
```

- Hľadanie najmenšej hodnoty

Funkcia *hladanie_minima()* je na nájdenie najmenšiu prvku v matici. Vo funkcií je vnorený for cyklus do n. Kde pomocná premenná *double min* sa nastaví na nulu a následne sa porovnáva s ďalšími hodnotami z matice. Najmenšia hodnota sa nakoniec vráti.

```
double hladanie_minima(int n, double **array)
{
    int i, j;
    double min = array[0][0];
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (min > array[i][j])
            {
                min = array[i][j];
            }
        }
    }
    return min;
}
```

- Hľadanie najväčšej hodnoty

Funkcia *hladanie_maxima()* je na nájdenie najväčšieho prvku v matici. Vo funkcií je vnorený for cyklus do n. Kde pomocná premenná *double max* sa porovnáva s ďalšími hodnotami z matice. Najväčšia hodnota sa nakoniec vráti.

```
double hladanie_maxima(int n, double **array)
{
    int i, j;
    double max;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (max < array[i][j])
            {
                max = array[i][j];
            }
        }
    }
    return max;
}
```

- Absolútna hodnota

Funkcia *absolutna_hodnota()* je na výpočet absolútnej hodnoty. Použil som tu ternárny operátor. Funkcia je použitá v maine vo for cykle kde sa počíta súčet jednotlivých riadkov.

```
double absolutna_hodnota(double hodnota)
{
    return ((hodnota < 0) ? (-hodnota) : (hodnota));
}
```

Testovanie

V prípade zadania veľkosti matice 0 alebo 1. Program na takéto hodnoty akceptovať nebude a opätovne bude čakať na znovu ich zadanie.

```
Lukas Lechovic 07
Zadanie 5:
Rieste problem, ktory v stvorcovej matici zadaneho lubovolneho rozmeru N x N
najde riadok (riadky cislujte vo vystupe), ktoreho absolutna hodnota suctu
jeho prvkov je maximum alebo minimum v danej matici.

Zadajte velkost stvorcovej matice: 0
Zadajte velkost stvorcovej matice: 1
Zadajte velkost stvorcovej matice:
```

Dáme si veľkosť matice 3. Následne si môžeme vybrať či si maticu vyplníme sami alebo si necháme vygenerovať. Tu program akceptuje iba čísla 1 a 2.

```
Zadajte velkost stvorcovej matice: 3

(1) Prvky si zadate sami
(2) Prvky si nechate nahodne vygenerovat
Zadajte 1 alebo 2: 5

(1) Prvky si zadate sami
(2) Prvky si nechate nahodne vygenerovat
Zadajte 1 alebo 2: 0

(1) Prvky si zadate sami
(2) Prvky si nechate nahodne vygenerovat
Zadajte 1 alebo 2: 3

(1) Prvky si zadate sami
(2) Prvky si nechate nahodne vygenerovat
Zadajte 1 alebo 2:
```

Vyberieme si možnosť 1. Prvky do matice si zadáme sami.

```
Stvorcova matica o velkosti 3:

1. riadok
Prvok [1][1]: 5
Prvok [1][2]: 3
Prvok [1][3]: 18

2. riadok
Prvok [2][1]: -5
Prvok [2][2]: 48.5
Prvok [2][3]: 98

3. riadok
Prvok [3][1]: 21
Prvok [3][2]: -3.5
Prvok [3][3]: 45

Najmensia hodnota v matici je -5.00
Najvacsia hodnota v matici je 98.00

        SUCET                   MIN                   MAX
(1)     26.00           >       -5.00           <       98.00
(2)     141.50          >       -5.00           >       98.00
(3)     62.50           >       -5.00           <       98.00
```

(1) 5+3+18=26
(2) -5+48.5+98=141.5
(3) 21-3.5+45=62.5

Výsledné hodnoty sú správne.

Otestovanie porovnania:

```
        SUCET                   MIN                   MAX
(1)     10.00           >       5.00            =       10.00
(2)     15.00           >       5.00            >       10.00
```

Program si môžeme následne spustiť znova tu funguje už spomínaný princíp.
Zadáme 1 pre pokračovanie, veľkosť matice dáme 4 a 2 maticu si vygenerujeme.

```
Chcete program spustit znova?
(1) Ano
(2) Nie
Zadajte iba 1 alebo 2: 1
Zadajte velkost stvorcovej matice: 4

(1) Prvky si zadate sami
(2) Prvky si nechate nahodne vygenerovat
Zadajte 1 alebo 2: 2

Stvorcova matica o velkosti 4:
(1)     -40.54          -67.93          38.30           44.20
(2)     -37.59          -28.98          42.45           75.30
(3)     -81.55          14.62           -2.81           92.47
(4)     9.14            -43.02          3.77            -59.17

Najmensia hodnota v matici je -81.55
Najvacsia hodnota v matici je 92.47

        SUCET                   MIN                   MAX
(1)     25.98           >       -81.55          <       92.47
(2)     51.19           >       -81.55          <       92.47
(3)     22.72           >       -81.55          <       92.47
(4)     89.28           >       -81.55          <       92.47
```

(1) -40.54-67.93+38.30+44.20=25.98
(2) -37.59-28.98+42.45+75.30=51.19
(3) -81.55+14.62-2.81+92.47=22.72
(4) 9.14-43.02+3.77-59.17=89.28

Výsledné hodnoty sú správne.
Program môžeme ukončiť stlačením 2.

```
Chcete program spustit znova?
(1) Ano
(2) Nie
Zadajte iba 1 alebo 2: 2
Dovidenia...
```

Vlastnosti programu:

User-friendly

Program má jednoduchý vzhľad bez akýchkoľvek rozptyľujúcich prvkov. Program rozumne a priamo komunikuje s používateľom aj z takým, ktorý ma iba základne počítačové znalosti. V prípade zadaní nesprávneho vstupu od používateľa program nespadne a opätovne čaká na znovu jeho zadanie.

Spoľahlivosť

Program bol niekoľko krát otestovaný mnou ako autorom ale aj viacerými ľuďmi pre získanie lepšej spätnej väzby. Akýkoľvek bug alebo odporúčanie bolo implementované.

Efektívnosť

Program je pamäťovo nenáročný. V prípade nedostatku pamäte, sa objaví chybové hlásenie. Program je svižný, za krátky časový úsek za minimálnu spotrebu pamäte dokáže spracovať údaje zo 100% spoľahlivosťou.

Celkové zhodnotenie

Tento program bol pre mňa veľkou výzvou, snažil som sa použiť všetko čo som vedel a mohlo by to programu ako takému pomôcť, pri vytváraní som sa veľa toho naučil. Program je spravený tak aby bol zrozumiteľný aj pre bežných používatelou PC.