НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЛАБОРАТОРНА РОБОТА №9

Структури

Виконала:
студентка групи ІПЗ-21007б
Соколовська Софія

Київ – 2022

1) Завдання:

Фермер займається вирощуванням буряків та веде щоденник, в який записує таку інформацію: рік, площа посівів (гектари), загальна маса зібраного врожаю (тони). Написати програму, що дозволяє ввести інформацію для кожного року та порахувати середню врожайність гектару землі по роках та в цілому за весь період. Представити інформацію у двох розрізах: відсортовану по зростанню року та відсортовану по спаданню продуктивності.

2) Код прграми:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct DiaryRecord {
   int year;
   float mass;
   float area;
   float fertility;
};

//returns random value from interval [lower, upper]
int getRandomValue(int lower, int upper) {
   return (rand() % (upper - lower + 1)) + lower;
}

//Initializess our dictionary with some random values.
void initDiary(struct DiaryRecord diary[], int diarySize) {
   srand(time(NULL));
   for (int i = 0; i < diarySize; i++) {
      //Even though we use random function here still we can have duplicates of the year.
      //Anyway it's not a big deal. Ñonsider that as multiple harvests. :)
      diary[i].year = getRandomValue(2000, 2022);
      diary[i].mass = getRandomValue(1, 100);
      diary[i].area = getRandomValue(1, 100);
      diary[i].fertility = diary[i].mass / diary[i].area;
   }
}

void printDiary(struct DiaryRecord diary[], int diarySize) {
   for (int i = 0; i < diarySize; i++) {
      printf("Year: %d; ", diary[i].year);
      printf("Mass: %.2f; ", diary[i].mass);
      printf("Area: %.2f; ", diary[i].area);
```

```
        printf("Fertility: %.2f", diary[i].fertility);
        printf("\n");
    }
}

//Swaps two diary records
void swap(struct DiaryRecord* r1, struct DiaryRecord* r2)
{
    struct DiaryRecord temp = *r1;
    *r1 = *r2;
    *r2 = temp;
}

//Last parameter determines by which field we will sort our diary.
//(0 - by year; 1 - by fertility)
void bubbleSort(struct DiaryRecord diary[], int diarySize, int sortingByField)
{
    for (int i = 0; i < diarySize - 1; i++) {
        for (int j = 0; j < diarySize - i - 1; j++) {
            //sort by year
            if (sortingByField == 0) {
                //Comparison sign determines sorting order. Ascending order here.
                if (diary[j].year > diary[j + 1].year) {
                    swap(&diary[j], &diary[j + 1]);
                }
            //sort by fertility
            } else if (sortingByField == 1) {
                // and Descending order here.
                if (diary[j].fertility < diary[j + 1].fertility) {
                    swap(&diary[j], &diary[j + 1]);
                }
            }
        }
    }
}

float getOverallAvgFertility(struct DiaryRecord diary[], int diarySize) {
    float totalFertility = 0;
    for (int i = 0; i < diarySize; i++) {
        totalFertility += diary[i].fertility;
    }

    return totalFertility / diarySize;
}
```

```
int main()
{
    //Just hardcoded amount of records in diary.
    int diarySize = 5;
    //Here is our diary. It's an array of records.
    struct DiaryRecord diary[diarySize];

    initDiary(diary, diarySize);
    printf("Initial diary (unsorted): \n");
    //print our initial Diary;
    printDiary(diary, diarySize);
    printf("Average fertility for entire period (%d entries in Diary): %.2f\n\n",
diarySize, getOverallAvgFertility(diary, diarySize));

    printf("Sorted by year (ascending): \n");
    bubbleSort(diary, diarySize, 0);
    printDiary(diary, diarySize);

    printf("Sorted by fertility (descending): \n");
    bubbleSort(diary, diarySize, 1);
    printDiary(diary, diarySize);

    return 0;
}
```

3) Результат програми:

```
Initial diary (unsorted):
Year: 2002; Mass: 30.00; Area: 29.00; Fertility: 1.03
Year: 2010; Mass: 74.00; Area: 12.00; Fertility: 6.17
Year: 2018; Mass: 76.00; Area: 83.00; Fertility: 0.92
Year: 2008; Mass: 36.00; Area: 68.00; Fertility: 0.53
Year: 2016; Mass: 47.00; Area: 24.00; Fertility: 1.96
Average fertility for entire period (5 entries in Diary): 2.12

Sorted by year (ascending):
Year: 2002; Mass: 30.00; Area: 29.00; Fertility: 1.03
Year: 2008; Mass: 36.00; Area: 68.00; Fertility: 0.53
Year: 2010; Mass: 74.00; Area: 12.00; Fertility: 6.17
Year: 2016; Mass: 47.00; Area: 24.00; Fertility: 1.96
Year: 2018; Mass: 76.00; Area: 83.00; Fertility: 0.92
Sorted by fertility (descending):
Year: 2010; Mass: 74.00; Area: 12.00; Fertility: 6.17
Year: 2016; Mass: 47.00; Area: 24.00; Fertility: 1.96
Year: 2002; Mass: 30.00; Area: 29.00; Fertility: 1.03
Year: 2018; Mass: 76.00; Area: 83.00; Fertility: 0.92
Year: 2008; Mass: 36.00; Area: 68.00; Fertility: 0.53
```

```
Initial diary (unsorted):
Year: 2018; Mass: 85.00; Area: 77.00; Fertility: 1.10
Year: 2015; Mass: 39.00; Area: 29.00; Fertility: 1.34
Year: 2005; Mass: 66.00; Area: 95.00; Fertility: 0.69
Year: 2011; Mass: 53.00; Area: 38.00; Fertility: 1.39
Year: 2017; Mass: 60.00; Area: 79.00; Fertility: 0.76
Average fertility for entire period (5 entries in Diary): 1.06

Sorted by year (ascending):
Year: 2005; Mass: 66.00; Area: 95.00; Fertility: 0.69
Year: 2011; Mass: 53.00; Area: 38.00; Fertility: 1.39
Year: 2015; Mass: 39.00; Area: 29.00; Fertility: 1.34
Year: 2017; Mass: 60.00; Area: 79.00; Fertility: 0.76
Year: 2018; Mass: 85.00; Area: 77.00; Fertility: 1.10
Sorted by fertility (descending):
Year: 2011; Mass: 53.00; Area: 38.00; Fertility: 1.39
Year: 2015; Mass: 39.00; Area: 29.00; Fertility: 1.34
Year: 2018; Mass: 85.00; Area: 77.00; Fertility: 1.10
Year: 2017; Mass: 60.00; Area: 79.00; Fertility: 0.76
Year: 2005; Mass: 66.00; Area: 95.00; Fertility: 0.69
```