ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ - СОФИЯ

Алгоритми и структури данни

лектор: доц. д-р Атанас Атанасов
Катедра “Информатика”

Лекция 6

СОРТИРОВКА НА ДАННИ

СОРТИРОВКА

Сортировката е метод за подреждане или пренареждане на елементите на дадено множество в определен ред.

Например във възходящ или низходящ ред.

Елементите на множеството може да са числа, но може и да са записи (структури от данни), съдържащи числова, текстова и/или друга информация.

Тогава сортировката може да е свързана, не само с подреждане на числа, но и с подреждане по азбучен ред на текстовата информация.

Сортировката се използва за различни цели:

- Азбучни указатели
- Сортиране по успех, по ЕГН и др.
- Картотеки на книги в библиотека
- Подреждане на информацията в бази данни
- Подреждане на масиви или файлове

Алгоритми за сортиране

Известни са множество алгоритми за сортировка. Някои по-известните, разгледани по долу са:

Сортиране чрез селекция
- метод на пряка селекция;

Сортиране чрез размяна:
- метод на мехурчетата (BubleSort);
- метод на шейкър сортиране “чрез клатене” (ShakerSort));
- метод на бърза сортировка на (Quicksort).

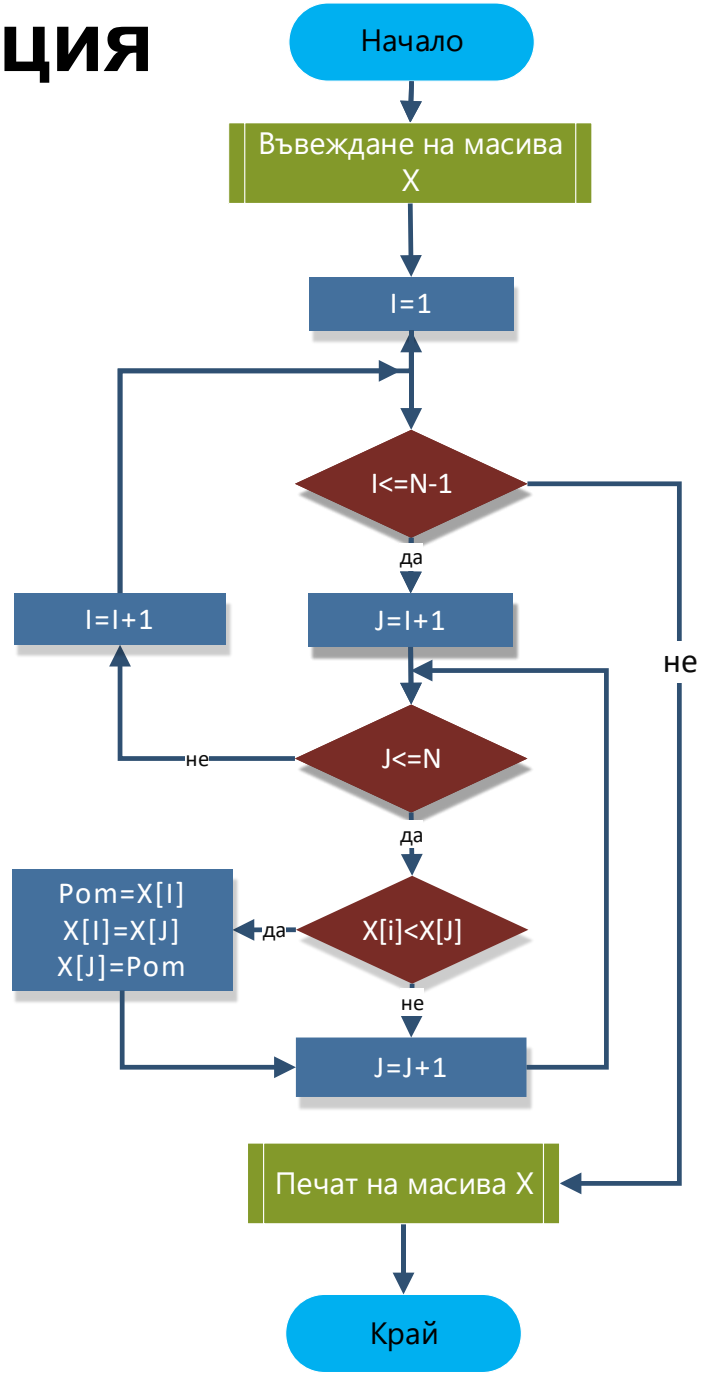
Сортиране чрез вмъкване:
- метод на пряко вмъкване;

Сортировка с пряка селекция

Пример 1:

Да се състави програма за сортиране на елементите на даден масив по метода на пряката селекция.

31 34 12 22 11	Най-малкият елемент е 11. Разменят се 11 и 31.
11 34 12 22 31	Най-малкият оставащ елемент е 12. Разменят се 12 и 34.
11 12 34 22 31	Вече имаме една част от списъка, която е сортирана. Сега се разменят 22 и 34.
11 12 22 34 31	Разменят се 31 и 34.
11 12 22 31 34	Списъкът е сортиран



Сортировка с пряка селекция

Пример 1:

Да се състави програма за сортиране на елементите на даден масив по метода на пряката селекция.

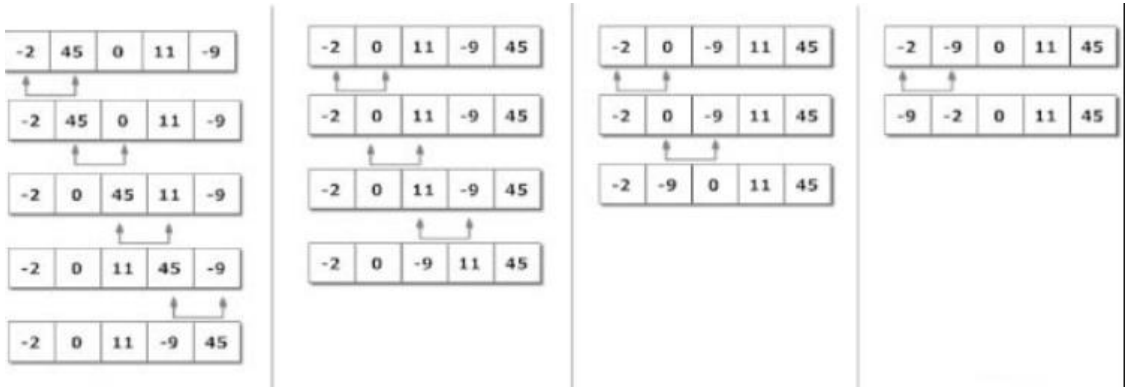
```
#include <iostream.h>
int main(){
     double x[200];  double pom;
     int i, j, n;
     cout<<"Broj na elementite na masiva n<=200: ";
     cin>>n;
     for(i=0;i<n;i++ ){
       cout<<"Vavedete X["<<i<<"] = ";  cin>>x[i];
      }
// sortirovka na masiva
for(i=0;i<n-1;i++ )
  for (j=i+1; j<n; j++)
           if(x[i]>x[j])   // ako se obyrne sravnenieto na <
              { pom = x[i];
                x[i]=x[j];
                x[j]=pom;
               }
 for(i=0;i<n;i++ ){  // izvevdane na sortirania masiv
       cout<<"X["<<i<<"] = "<<x[i]<<endl;
      }return 0;
}
```

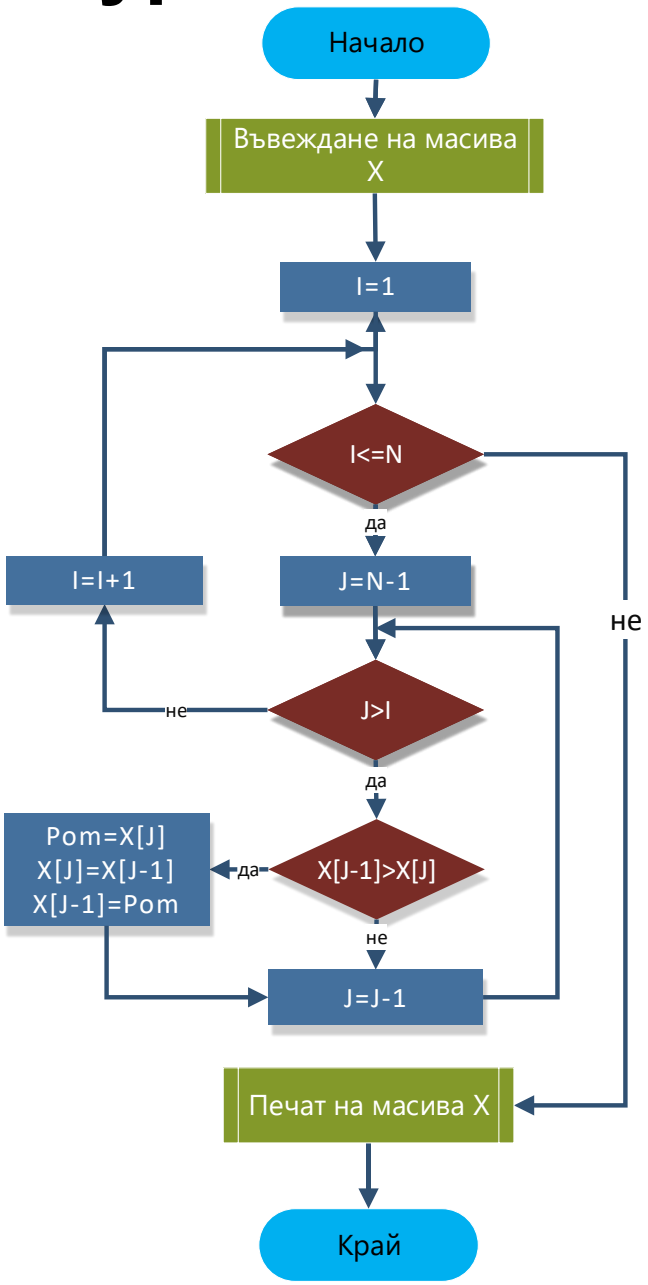
Сортировка по метода на мехурчетата

Пример 2:

Да се състави програма за сортиране на елементите на даден масив по метода на мехурчетата.

6 5 3 1 8 7 2 4





Сортировка по метода на мехурчетата

Пример 2:

Да се състави програма за сортиране на елементите на даден масив по метода мехурчетата.

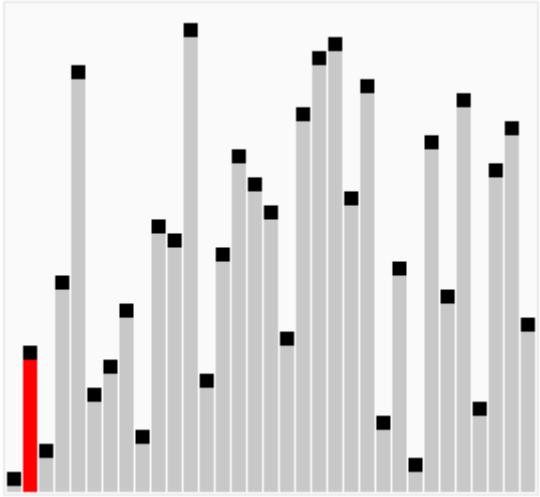
```
#include <iostream.h>
int main(){
     double x[200];  double pom;
     int i, j, n;
     cout<<"Broj na elementite na masiva n<=200: ";
     cin>>n;
     for(i=0;i<n;i++ ){
       cout<<"Vavedete X["<<i<<"] = ";  cin>>x[i];
      }
// sortirovka na masiva
for(i=0;i<n-1;i++ )
  for (j=n-1; j>i; j--)
          if(x[j-1]>x[j])   // ako se obyrne sravnenieto na <
             { pom = x[j];
               x[j]=x[j-1];
               x[j-1]=pom;
              }
 for(i=0;i<n;i++ ){  // izvevdane na sortirania masiv
       cout<<"X["<<i<<"] = "<<x[i]<<endl;
      }return 0;
}
```

Сортировка по метода на разклащането

Пример 3:

Да се състави програма за сортиране на елементите на даден масив по метода на разклащането.

```
k=n;  R=n-1; L=1;
do{
      for (j=R; j>=L; j--)
           if (x[j-1]>x[j])       { pom=x[j];   x[j]=x[j-1];   x[j-1]=pom; k=j; } //надясно
       L=k+1;
       for (j=L; j<=R; j++)
           if (x[j-1]>x[j])       { pom=x[j];   x[j]=x[j-1];    x[j-1]=pom; k=j; } //наляво
       R=k-1;
} while (L<R);
```



Сортировка по метода на разклащането

Пример 3:

Да се състави програма за сортиране на елементите на даден масив по метода на вмъкването.

```
#include <iostream.h>
int main(){
     double x[200];  double pom;
     int i, j, n;
     cout<<"Broj na elementite na masiva n<=200: ";   cin>>n;
     for(i=0;i<n;i++ ){
        cout<<"Vavedete X["<<i<<"] = ";  cin>>x[i];     }
k=n; R=n-1; L=1; // sortirovka na masiva
do{
for (j=R; j>=L; j--)
   if (x[j-1]>x[j]) { pom=x[j];  x[j]=x[j-1];  x[j-1]=pom; k=j; }
L=k+1;
for (j=L; j<=R; j++)
   if (x[j-1]>x[j]) { pom=x[j];  x[j]=x[j-1];  x[j-1]=pom; k=j; }
R=k-1;
}
while (L<R);
 for(i=0;i<n;i++ ){  // izvevdane na sortirania masiv
        cout<<"X["<<i<<"] = "<<x[i]<<endl;
      }return 0;
}
```

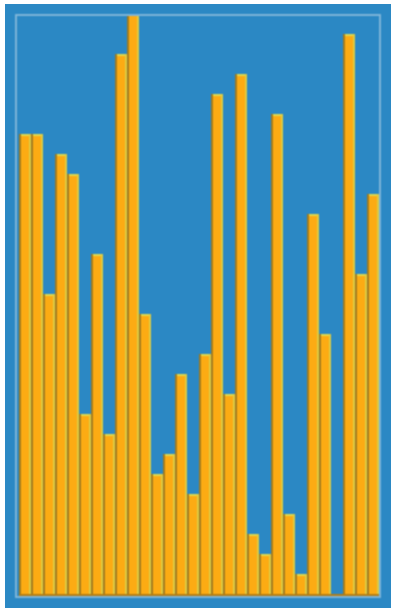
Сортировка по метода на вмъкването

Метод на вмъкването (Insertion Sort)

При този метод се правят N стъпки, като след i-тата стъпка са сортирани първите i елемента.

При първата стъпка са сортирани първите два елемента. На следващата стъпка са сортирани първите три, и така нататък, докато на последната стъпка са сортирани всички елементи.

На всяка нова i-та стъпка трябва да се постави "вмъкне" i-тия елемент във вече сортираните i-1. Но понеже тези елементи вече са сортирани, ако се местим от дясната страна на вече сортираната част на масива ще се разменят i-тия елемент с левите му елементи, докато отиде на мястото си, сиреч не е по-малък от левия си, защото останалите са вече сортирани.



6 5 3 1 8 7 2 4

Сортировка по метода на вмъкването

Пример 4:

Да се състави програма за сортиране на елементите на даден масив по метода на вмъкването.

```
#include <iostream.h>
int main(){
    double x[200];  double pom;
    int i, j, n;
    cout<<"Broj na elementite na masiva n<=200: ";   cin>>n;
    for(i=0;i<n;i++ ){
      cout<<"Vavedete X["<<i<<"] = ";  cin>>x[i];     }
// Insertion sort
for (i = 1; i <n ; i++) {
       for (j = i; j > 0  ; j--)
             if (x[j] < x[j - 1])
                  { pom=x[j];   x[j]=x[j-1];    x[j-1]=pom; }

   }
for(i=0;i<n;i++ ){  // izvevdane na sortirania masiv
       cout<<"X["<<i<<"] = "<<x[i]<<endl;
     }return 0;
}
```

Бързо сортиране Qucksort

Това е рекурсивен алгоритъм, който на всяка стъпка фиксира един от подадените елементи за разделител (*pivot*), след което разделя числата на три части - тези по-малки от разделителя, тези равни на него, и тези по-големи от него. Ако сортираме първата част, след нея поставим втората част (без да правим нищо върху нея - тя вече е сортирана, тъй като всички елементи в нея са с равна стойност), и накрая сортираме третата част и я поставим след първата и втората, в крайна сметка всички елементи ще са сортирани.

array[0]	aray<=Pivot		array[M-1]	Pivot	array[M+1]	array=>Pivot		array[N-1]
0	.....	M-2	M-1	M	M+1	M+2	.....	N-1

Така стъпките на алгоритъма (при всяко викане на рекурсията) са:

1. Избираме един от елементите за *pivot*.

2. Вземаме всички елементи, по-малки от *pivot* (да кажем *M1* на брой), сортираме ги прилагайки същия алгоритъм рекурсивно, и ги поставяме на позиции 0...*M-1* от резултатния масив.

3. Вземаме всички елементи, равни на *pivot* (да кажем *M2* на брой) и ги поставяме на позиции *M+1*, *M+2..N-1* от резултатния масив.

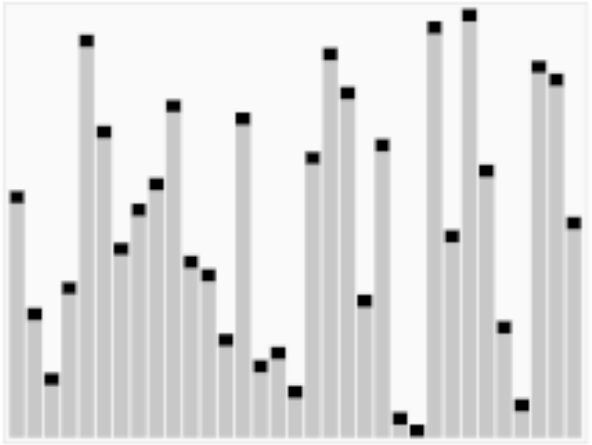
4. Вземаме всички елементи, по-големи от *pivot* (да кажем *M3* на брой), сортираме ги прилагайки същия алгоритъм рекурсивно, и ги поставяме на позиции *M1+M2*...*N*-1 от резултатния масив.

Бързо сортиране Quicksort

Пример 5:

Да се състави програма за сортиране на елементите на даден масив по метода на бързото сортиране.

По-долу е дадена рекурсивната функция quicksort(), която се извиква от главната програма с параметри указател към масива за сортиране и брой елементи на масива. Функцията се извиква рекурсивно докато размера size стане равен на 1 или 0.



```
void quicksort(int* array, int size) {
    if (size > 1) {
        int pivot = array[random() % size];

        // Less
        int left = 0;
        for (int i = 0; i < size; i++)
            if (array[i] < pivot) swap(array[i], array[left++]);
        quicksort(array, left);

        // Greater
        int right = size - 1;
        for (int i = size - 1; i >= left; i--)
            if (pivot < array[i]) swap(array[i], array[right--]);
        quicksort(array + right + 1, size - right - 1);
    }
}
```

Сортиране на масив от структури

Да се състави програма за въвеждане на данни за студенти в масив от записи, имащи следната структура: име, факултетен номер, специалност и среден успех . Да се изчисли средният успех на студентите. Да се направи сортиране и извеждане на данните за студентите по успех в низходящ ред.

За решаване на задачата е дефиниран тип структура с име stud. Дефиниран е също масив от 50 елементи от тип структура - student[50]. В този масив се въвеждат данните за студентите. Достъпът до полето name за i –тия студент се извършва по следния начин: student[i].name. Аналогично се извършва достъпа и до останалите полета. Сортирането на записите се извършва по метода на най-малкия елемент, като се сравняват полетата uspeh за всеки запис.

Структура – Пример

```
#include <iostream.h>
int main(){
struct stud
{           char   name[30];
            char   facNumber[12];
            char   specialnost[20];
            float  uspeh;
};
stud student[50], pom;
 int n;// dejstvitelen broj na studentite
 int i;
 cout<<" Vavedete broj na studentite n <= 50 :";  cin>>n;
 for(i=0;i<n;i++)
             {cout<<"Vavedete ime na student:";
              cin>>student[i].name;
              cout<<"Vavedete facNumber:";
              cin>>student[i].facNumber;
              cout<<"Vavedete specialnost:";
              cin>>student[i].specialnost;
              cout<<"Vavedete uspeh:";
              cin>>student[i].uspeh;
}
```

Структура – Пример продължение

```
// сортиране по успех
int j;
  for(i=0;i<n-1;i++)
              for(j=i+1;j<n;j++)
                        if(student[i].uspeh < student[j].uspeh)
                                  {pom=student[i];
                                   student[i]= student[j];
                                   student[j] = pom;
                                  }
// извеждане на сортирания масив със студентите
  for(i=0;i<n;i++)
              {cout<<" ime:            "<<student[i].name<<endl;
               cout<<" facNumber:  "<<student[i].facNumber<<endl;
               cout<<" specialnost:   "<<student[i].specialnost<<endl;
               cout<<" uspeh:          "<<student[i].uspeh<<endl<<endl;
              }
  return 0;
 }
```