Digitální učební materiál

Číslo projektu:	CZ.1.07/1.5.00/34.0548
Název školy:	Gymnázium, Trutnov, Jiráskovo náměstí 325
Název materiálu:	VY_32_INOVACE_149_IVT
Autor:	Ing. Pavel Bezděk
Tematický okruh:	Algoritmy
Datum tvorby:	srpen 2013
Ročník:	4. ročník a oktáva
Anotace:	Algoritmus IX. – Třídění dat I. - Insert Sort, Select Sort
Metodický pokyn:	Při výuce nutno postupovat individuálně.

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Autor	**Ing. Pavel Bezděk**		
Vytvořeno dne	**6. 8. 2013**		
Odpilotováno dne	**17. 2. 2014**	**ve třídě**	**8.Y**
Vzdělávací oblast	**Informatika a informační a komunikační technologie**		
Vzdělávací obor	**Informatika a výpočetní technika**		
Tematický okruh	**Algoritmus**		
Téma	**Algoritmus IX. - Třídění dat I. - Insert Sort, Select Sort**		
Klíčová slova	**Algoritmus, třídění dat, Insert Sort, Select Sort**		

Třídění dat

Insert Sort a Select Sort

Třídění přímým vkládáním
INSERT SORT Pascal

```
program Prime_Vkladani;
uses CRT;  const N = 10;
type Pole = array[1..N] of integer;   var i:integer;  A:Pole;
 procedure PrimeVkladani(var A: Pole); {začátek deklarace procedury
PrimeVkladani}
var i,j: integer;             {indexy prvku}
 X: integer;              {pro výměnu prvku}
Hledat:Boolean;
begin  for i:=2 to N do          {zatřiďujeme číslo z pozice i}
  begin    X:=A[i];    j:=i-1;
  Hledat:=X<A[j];    while Hledat do           {hledáni správné pozice}
    begin      A[j+1]:=A[j];      j:=j-1;      if j=0 then Hledat:=false
                                                 else      Hledat:=X<A[j]
    end;
  A[j+1]:=X
   end
end;  {konec deklarace procedury PrimeVkladani}
begin
writeln('Zadej ', N, ' netříděných čísel:');                  for i:=1 to N do read(A[i]);
PrimeVkladani(A);     {volání procedury PrimeVkladani}
writeln('Setříděno:');    for i:=1 to N do write(A[i]:5);    repeat until keypressed;
end.
```

Třídění přímým vkládáním
INSERT SORT C++

```
#include <iostream>
//#include <conio.h>
const int POCET=10;
void insert(int p[],int n);
using namespace std;
int main()
{
        int a[POCET]={8,3,7,1,10,5,6,4,2,9};     // pole tridenych prvku
        int pole[POCET];                          // vstupní pole pro trideni
        int pole3[POCET+1];                       //  Pole pro insertsort
        int pocet=POCET;                  // vstupni parametr - pocet prvku
        int k;
        for (k=0; k<pocet; k++)           // pole tridenych prvku presuneme
                pole[k]=a[k];             // do vstupniho pole pro trideni
    cout<< "Prvky pole, ktere mame setridit: "<<endl;
        for (k=0; k<pocet; k++)           // cyklus pro zobrazeni vysledku
                 cout<< pole[k]<<"  ";
        for (k=0; k<pocet; k++)           //  pole tridenych prvku presuneme
                pole3[k+1]=a[k];          //  do vstupniho pole pro trideni
                cout<<endl;
```

```
insert(pole3,pocet);                        // zde pouziji o 1 delsi pole3
        cout<<("Setridene pole (insertsort): \n");
        for (k=1; k<=pocet; k++)            // cyklus pro zobrazeni vysledku
                cout<< pole3[k]<<"  " ;
                cout<< ("\n");              // zalomi radek po vytisteni vysledku
        for (k=0; k<pocet; k++)
                pole[k]=a[k];
        return 0;
}
```

```
void insert(int p[],int n)                    // trideni vkladanim - insertsort
                        // zde jsou tridene prvky ulozeny v poli od indexu 1
                        // p[0] plni ulohu zarazky
{
int i,j;
for (i=2;i<=n;i++)
            {
            p[0] = p[i]; j = i-1;
            while (p[0] < p[j])
                        {
                                    p[j+1] = p[j]; j--;
                        }
                        p[j+1] = p[0];
            }
}
```

Schéma třídění INSERT SORT - přímé vkládání

II.	45	56	13	43	95	19	7	68
prvek	45	56	13	43	95	19	7	68
pole	45	56	13	43	95	19	7	68

	nesetříděná část pole
	porovnávané prvky
	setříděná část pole

III.	45	56	13	43	95	19	7	68
prvek	45	56	13	43	95	19	7	68
pole	45	13	56	43	95	19	7	68
	13	45	56	43	95	19	7	68

IV.	13	45	56	43	95	19	7	68
prvek	13	45	56	43	95	19	7	68
pole	13	45	43	56	95	19	7	68
	13	43	45	56	95	19	7	68
	13	43	45	56	95	19	7	68

V.	13	43	45	56	95	19	7	68
prvek	13	43	45	56	95	19	7	68
pole	13	43	45	56	95	19	7	68
	13	43	45	56	95	19	7	68
	13	43	45	56	95	19	7	68
	13	43	45	56	95	19	7	68

VI.	13	43	45	56	95	19	7	68
prvek	13	43	45	56	95	19	7	68
pole	13	43	45	56	19	95	7	68
	13	43	45	19	56	95	7	68
	13	43	19	45	56	95	7	68
	13	19	43	45	56	95	7	68
	13	19	43	45	56	95	7	68

	nesetříděná část pole
	porovnávané prvky
	setříděná část pole

VII.	13	19	43	45	56	95	7	68
prvek	13	19	43	45	56	95	7	68
pole	13	19	43	45	56	7	95	68
	13	19	43	45	7	56	95	68
	13	19	43	7	45	56	95	68
	13	19	7	43	45	56	95	68
	13	7	19	43	45	56	95	68
	7	13	19	43	45	56	95	68

	nesetříděná část pole
	porovnávané prvky
	setříděná část pole

VIII.	7	13	19	43	45	56	95	68
prvek	7	13	19	43	45	56	95	68
pole	7	13	19	43	45	56	68	95
	7	13	19	43	45	56	68	95
	7	13	19	43	45	56	68	95
	7	13	19	43	45	56	68	95
	7	13	19	43	45	56	68	95
	7	13	19	43	45	56	68	95
	7	13	19	43	45	56	68	95

7	13	19	43	45	56	68	95

Časová a paměťová složitost přímého vkládání – Insert Sort

Začnu 2. prvkem v poli, porovnám ho s 1. prvkem poli (s prvkem s hodnotou indexu o 1 menší), pokud je prvek s vyšším indexem menší, prvky prohodím. Pokud je prvek s vyšším indexem větší nebo roven, nic neprovedu.

Pak vezmu 3.prvek v poli a porovnám s 2. prvkem (s prvkem s hodnotou indexu o 1 menší), po porovnání a případném prohození, porovnám 2. prvek s 1.prvkem.

Vezmu 4. prvek a porovnám s 3. prvkem, pak 3. prvek s 2. prvkem, 2. prvek s 1. prvkem. Na začátku pole se mi vytváří setříděná část pole, do které postupně vkládám prvky z nesetříděné části pole (na opačné straně pole).

Pak vezmu 5. prvek a porovnám s 4. prvkem, ……., až 2. a 1. prvek.

Nakonec beru N. prvek a porovnávám s (N-1). prvkem , (N-1) a (N-2) prvek, až se dostanu k porovnání 2. a 1. prvku

Složitost : nejlepší př.: $C= (N-1)$ $M=2(N-1)$ **C: časová složitost**

průměr. př.: $C= (N^2-N-2)/4$ $M=(N^2-9N-10)/4$ **M: paměťová složitost**

nejhorší př. : $C= (N^2-N)/2-1$ $M=N^2+3N-4)/2$

Asymptotická složitost :

časová $O(n^2)$ - **nejhorší a prům. př.**; $O(n)$ - **nejlepší př.**

paměťová $O(n^2)$ - **nejhorší a prům. př.**; $O(n)$ - **nejlepší př.**

Když jsou data již správně setříděná (od nejmenší hodnoty k největší), není tedy již co třídit, je složitost časová i paměťová lineární.

Třídění - přímý výběr - Select Sort Pascal

```
program Primy_Vyber;
uses CRT; const N = 10;-
type Pole = array[1..N] of integer; var i:integer;    A:Pole;
procedure PrimyVyber(var A: Pole); {Zacatek deklarace procedury
PrimyVyber}
var i,j,k: integer;                    {indexy prvku}
 X: integer;                          {pro vymenu prvku}
 begin  for i:=1 to N-1 do           {umistit cislo na pozici i}
   begin
      k:=i;
      for j:=i+1 to N do if A[j] < A[k] then k:=j;  {vyhledani minima}
      if k > i then          {vymena prvku s indexy i, k}
                  begin X:=A[k]; A[k]:=A[i]; A[i]:=X end
    end
 end;  {Konec deklarace procedury PrimyVyber}
begin   {Telo programu}
writeln('Zadej ', N, ' netridenych cisel:');  for i:=1 to N do read(A[i]);
PrimyVyber(A); {Volani procedury PrimyVyber}
writeln('Setrideno:');  for i:=1 to N do write(A[i]:5);
writeln; {Tisk jiz setrideneho pole}                repeat until keypressed;
end.
```

```
#include <iostream>
using namespace std;
const int index=10;
int a[index];                          // pole tridenych prvku
int pole[index];                       //  vstupní pole pro trideni
void select(int p[],int n);            //  prototyp funkce select

int ii;
int main()
{
   cout<<"Zadej prvky pole"<<endl;
 for (ii=0;ii<index;ii++)
 {   cout<<"Zadej "<<ii<<". prvek pole:";
     cin >>a[ii];
 }
```

```
cout<< endl;
 cout<<"Zadali jste tyto prvky pole: "<<endl;
 for (ii=0;ii<index;ii++)
 {   cout<<ii<<". prvek pole: "<<a[ii]<<endl;
 }
   cout << endl;
            int pocet;                                  // nebo    int pocet=index;
            pocet=index;                                // vstupni parametr - pocet prvku
            int kk;
            for (kk=0; kk<pocet; kk++)                  // pole tridenych prvku presuneme
                        pole[kk]=a[kk];                 // do vstupniho pole pro trideni

            select(pole,pocet);                         // zavolani funkce select

                        cout<<endl;


            return 0;
}
```

```
void select(int p[],int n)          // trideni pole primym vyberem - selectsort
{
        int i,j,k;
        int x;
        for (i=0; i<n-1; i++)
        {
                k=i; x=p[i];
                for (j=i+1; j<n; j++)
                        if (p[j]<x)
                        {
                                k=j; x =p[j];
                        }
                p[k]=p[i];p[i]=x;          // na i-te misto se dostal nejmensi prvek

        }                                  //z mnoziny prvku i+1-ho az n-teho
        cout<<"Setridene pole (selectsort):"<<endl;
                for (i=0; i<n; i++) { cout<<i<<". prvek pole: "<<p[i]<<endl; }
   cout<<endl;
}
```

Schéma třídění SELECT SORT

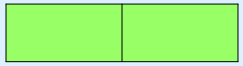
	45	56	13	43	95	19	7	68
I.	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	45	56	13	43	95	19	7	68
	7	56	13	43	95	19	45	68

	7	56	13	43	95	19	45	68
II.	7	56	13	43	95	19	45	68
	7	56	13	43	95	19	45	68
	7	56	13	43	95	19	45	68
	7	56	13	43	95	19	45	68
	7	56	13	43	95	19	45	68
	7	56	13	43	95	19	45	68
	7	56	13	43	95	19	45	68
	7	13	56	43	95	19	45	68

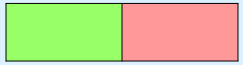
	7	13	56	43	95	19	45	68
III.	7	13	56	43	95	19	45	68
	7	13	56	43	95	19	45	68
	7	13	56	43	95	19	45	68
	7	13	56	43	95	19	45	68
	7	13	56	43	95	19	45	68
	7	13	56	43	95	19	45	68
	7	13	19	43	95	56	45	68

nesetříděná část pole

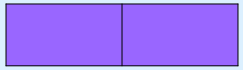
setříděná část pole



porovnávané prvky pole



porovnávané prvky pole a vznik nového minima



výměna 1. prvku ještě nesetříděného pole se nalezeným minimem v nesetříděném poli

min. prvek v nesetříd. poli

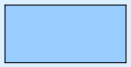
	7	13	19	43	95	56	45	68
IV.	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68

	7	13	19	43	95	56	45	68
V.	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	95	56	45	68
	7	13	19	43	45	56	95	68

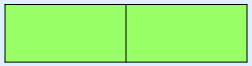
	7	13	19	43	45	56	95	68
VI.	7	13	19	43	45	56	95	68
	7	13	19	43	45	56	95	68
	7	13	19	43	45	56	95	68
	7	13	19	43	45	56	95	68

	7	13	19	43	45	56	95	68
VII.	7	13	19	43	45	56	95	68
	7	13	19	43	45	56	95	68
	7	13	19	43	45	56	68	95
VIII.	7	13	19	43	45	56	68	98

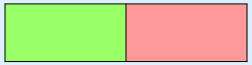
nesetříděná část pole



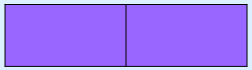
setříděná část pole



porovnávané prvky pole



porovnávané prvky pole a vznik nového minima



výměna 1. prvku ještě nesetříděného pole se nalezeným minimem v nesetříděném poli

min. prvek v nesetříd. poli

Časová a paměťová složitost přímého výběru SELECT SORT

V poli 1..N najdu min. prvek a prohodím s prvním prvkem v poli.

Pokud je min. prvek současně prvkem určeným k prohození (zde 1.prvek), nic neprovedu

V poli 2..N najdu min. prvek a prohodím s druhým prvkem v poli

V poli 3..N najdu min. prvek a prohodím s druhým prvkem v poli

.

.

V poli (N-1)..N najdu min. prvek a prohodím s předposledním prvkem v poli

		C:časová složitost	M: paměťová složitost
Složitost :	**nejlepší př.:**	$C= (N^2-N)/2$	$M=3(N-1)$
	průměr. př.:	$C= (N^2-N)/2$	$M=N(\ln N+0,57)$
	nejhorší př.:	$C= (N^2-N)/2;$	$M=N^2/4+3(N-1)$

Asymptotická složitost :

časová $O(n^2)$ $\Omega(n^2)$

paměťová $O(n)$ **- nejlepší** $O(n*\ln n)$ **- průměrná** $O(n^2)$ **– nejhorší**

Přímý výběr zabere vždy $O(n^2)$ času, protože při výběru prvku v sekvenci musíme danou sekvenci projít vždy celou.

Doporučená videa

Insert Sort:

http://www.youtube.com/watch?v=ROalU379l3U

Select Sort:

http://www.youtube.com/watch?v=Ns4TPTC8whw

Použité zdroje

WIRTH, Niklaus. *Algoritmy a štruktúry údajov*. 2.vyd. Bratislava: Alfa, 1989, 481 s. ISBN 80-050-0153-3.