Laporan Praktikum Algoritma Struktur Data

Bab Sorting

1. **Menu 6 Metode Tipe Data Integer**
2. Listing Program

#include<stdio.h>

#include<stdlib.h>

#define MAKS 10

void buble(int []);

void insertion(int []);

void selection (int []);

void tampil(int[]);

void tukar(int \*,int\*);

void shell(int [] );

void menu(int [] );

void backup(int [], int [] );

void copy(int [], int []);

void merge(int [], int, int, int, int);

void mergeRekursif(int [], int, int, int);

void mergeAwal(int [], int , int );

void quick(int [], int , int );

void quickRekursif(int [], int, int, int);

int partition(int [], int, int, int);

int subMenu();

int n;

main()

{

int A[MAKS], B[MAKS], i;

printf("Masukkan Banyak Data : ");

scanf("%d", &n);

puts("");

for (i=0;i<n;i++){

printf("Masukkan data ke %d : ",i+1);

scanf("%d",&A[i]);

}

backup(A,B);

do{

copy(A,B);

puts("");

menu(A);

}while (1);

}

void menu(int a[]){

int pil;

puts("==========================================");

puts("Menu Sorting 6 Metode Data Integer");

puts("==========================================");

puts("1. Insertion");

puts("2. Selection");

puts("3. Bubble");

puts("4. Shell");

puts("5. Merge");

puts("6. Quick");

puts("7. Exit\n");

printf("Masukkan Pilihan Anda : ");

scanf("%d", &pil);

switch (pil){

case 1:

insertion(a);

break;

case 2:

selection (a);

break;

case 3:

buble(a);

break;

case 4:

shell(a);

break;

case 5:

mergeAwal(a, 0, n-1);

printf("\nData Setelah Sorting\n");

tampil(a);

break;

case 6 :

quick(a, 0, n-1);

printf("\nData Setelah Sorting\n");

tampil(a);

break;

case 7 : puts("Selesai....");

exit(0);

break;

default : puts("Pilihan menu tidak valid");

}

}

int subMenu(){

int pil;

puts("");

puts("Pilihan Mode");

puts("1. Ascending");

puts("2. Descending\n");

printf("Masukkan Pilihan Anda : ");

scanf("%d", &pil);

return pil;

}

void backup(int a[], int b[])

{

int i;

for (i=0;i<n;i++){

b[i]=a[i];

}

}

void copy(int a[], int b[]){

int i;

for(i=0;i<n;i++){

a[i]=b[i];

}

}

void shell(int k[])

{

int i=0, jarak=n, mode, kondisi,swap;

mode= subMenu();

while(jarak>1){

jarak=jarak/2;

swap=1;

while(swap==1){

i=0;

swap=0;

while(i<(n-jarak)){

if(mode==1)

kondisi = k[i]>k[jarak+i];

else

kondisi = k[i]<k[jarak+i];

if(kondisi){

tukar(&k[i],&k[jarak+i]);

swap=1;

}

i++;

}

}

}

printf("\nData Setelah Sorting\n");

tampil(k);

}

void buble(int k[])

{

int i=0, pos\_akhir,mode, kondisi, swap=1;

mode=subMenu();

pos\_akhir=n-2;

while(pos\_akhir>=0&&swap==1){

i=0;

swap=0;

while(i<=pos\_akhir){

if(mode==1)

kondisi = k[i]>k[i+1];

else

kondisi = k[i]<k[i+1];

if(kondisi){

tukar(&k[i],&k[i+1]);

swap=1;

}

i++;

}

pos\_akhir--;

}

printf("\nData Setelah Sorting\n");

tampil(k);

}

void insertion(int k[])

{

int i=1, j, key, mode, kondisi;

mode=subMenu();

while(i<n){

key=k[i];

j=i-1;

while(j>=0){

if(mode==1)

kondisi=k[j]>key;

else

kondisi=k[j]<key;

if(kondisi){

k[j+1]=k[j];

j=j-1;

}

else

break;

}

k[j+1]=key;

i++;

}

printf("\nData Setelah Sorting\n");

tampil(k);

}

void selection(int k[])

{

int i=0, j, min, kondisi, mode;

mode=subMenu();

while(i<n-1){

min=i;

j=i+1;

while(j<n){

if(mode==1)

kondisi = k[j]<k[min];

else

kondisi = k[j]>k[min];

if(kondisi){

min=j;

}j++;

}

tukar(&k[min],&k[i]);

i++;

}

printf("\nData Setelah Sorting\n");

tampil(k);

}

void mergeAwal(int A[], int l, int r){

int mode;

mode = subMenu();

mergeRekursif(A,l,r,mode);

}

void merge(int data[], int l, int m, int r, int mode){

int kiri1, kiri2, kanan1, kanan2, i, j, kondisi, hasil[MAKS];

kiri1=l;

kanan1=m;

kiri2=m+1;

kanan2=r;

i=l;

while((kiri1<=kanan1) && (kiri2<=kanan2)){

if(mode==1)

kondisi=data[kiri1]<=data[kiri2];

else

kondisi=data[kiri1]>data[kiri2];

if(kondisi){

hasil[i]=data[kiri1];

kiri1++;

} else {

hasil[i]=data[kiri2];

kiri2++;

}

i++;

}

while(kiri1<=kanan1){

hasil[i]=data[kiri1];

kiri1++;

i++;

}

while(kiri2<=kanan2){

hasil[i]=data[kiri2];

i++;

kiri2++;

}

j=l;

while(j<=r){

data[j]=hasil[j];

j++;

}

}

void mergeRekursif(int A[], int l, int r, int mode){

int m;

if(l<r){

m=(l+r)/2;

mergeRekursif(A,l, m, mode);

mergeRekursif(A,m+1,r, mode);

merge(A,l,m,r, mode);

}

}

void quick(int A[], int p, int r){

int mode;

mode=subMenu();

quickRekursif(A,p, r, mode);

}

void quickRekursif(int A[], int p, int r , int mode){

int q;

if(p<r){

q=partition(A,p,r, mode);

quickRekursif(A, p, q, mode);

quickRekursif(A, q+1, r, mode);

}

}

int partition(int A[], int p, int r, int mode){

int x, i, j;

x=A[p];

i=p;

j=r;

do{

if(mode==1){

while(A[j]>x)

j--;

while(A[i]<x)

i++;

} else {

while(A[j]<x)

j--;

while(A[i]>x)

i++;

}

if(i<j){

tukar(&A[i],&A[j]);

j--;

i++;

} else {

return j; }

tampil(A);

}while(1);

return j;

}

void tukar(int \*a,int \*b)

{

int temp;

temp=\*b;

\*b=\*a;

\*a=temp;

}

void tampil(int a[])

{

int i;

for (i=0;i<n;i++){

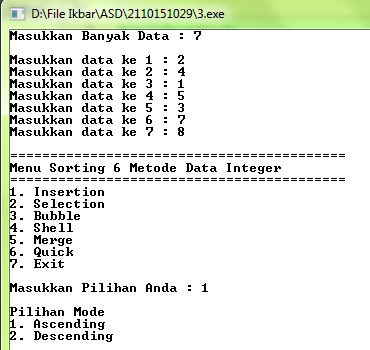
printf("%d ",a[i]);

}

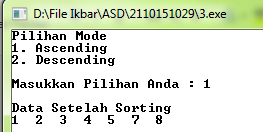
puts("");

}

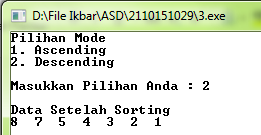
1. **Capture Output**



Tampilan Menu



Ascending



Descending

1. **Menu 6 Metode Tipe Data Struct**
2. Listing Program

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAKS 5

typedef struct data {

int no;

char nama[20];

int nilai;

}siswa;

void input(siswa [], int);

void backup(siswa [], siswa[], int);

void copy\_array(siswa [], siswa[], int);

void menu(siswa [], int);

void insertion(siswa [], int);

void selection(siswa [], int);

void bubble (siswa [], int);

void shell (siswa [], int);

void quick(siswa [], int, int);

void quickRekursif(siswa [], int, int, int, int);

int partition(siswa [], int , int , int, int);

void mergeAwal(siswa [], int , int );

void merge(siswa [], int, int, int, int, int);

void mergeRekursif(siswa [], int, int, int, int);

void tampil(siswa []);

void tukar(siswa \*, siswa \*);

int status, mode, n;

main(){

siswa data[MAKS], B[MAKS];

puts("-------------------------");

puts("Menu Sorting Data Struct");

puts("-------------------------\n");

printf("Masukkan Batas Data : "); scanf("%d", &n);

puts("-----------------------------");

input(data, n);

backup(data, B, n);

do{

copy\_array(data, B, n);

puts("");

menu(data, n);

}while(1);

}

void menu(siswa a[], int x){

int pil;

puts("Pilihan Menu\n");

puts("1. Insertion");

puts("2. Selection");

puts("3. Bubble");

puts("4. Shell");

puts("5. Merge");

puts("6. Quick");

puts("7. Exit\n");

printf("Input Menu : "); scanf("%d", &pil);

switch(pil){

case 1:

insertion(a, x);

break;

case 2:

selection(a, x);

break;

case 3:

bubble(a, x);

break;

case 4:

shell(a,x);

break;

case 5:

mergeAwal(a,0,n-1);

break;

case 6:

quick(a,0,n-1);

break;

case 7:

exit(0);

break;

default:

puts("Invalid Input.......");

break;

}

puts("");

}

int subMenu(){

int pil;

puts("");

puts("Pilihan Mode\n");

puts("1. Ascending");

puts("2. Descending\n");

printf("Masukkan Pilihan Anda : ");

scanf("%d", &pil);

return pil;

puts("\n");

}

int subMenu2(){

int pil;

puts("");

puts("Pilihan Mode\n");

puts("1. Sort By Number");

puts("2. Sort By Name");

puts("3. Sort By Nilai\n");

printf("Masukkan Pilihan Anda : ");

scanf("%d", &pil);

return pil;

puts("\n");

}

void input(siswa x[], int y){

int i;

for(i=0 ; i<y ; i++){

printf("Masukkan data ke-%d : \n", i+1);

printf("No : "); scanf("%d", &x[i].no);

printf("Nama : "); scanf("%s", &x[i].nama);

printf("Nilai : "); scanf("%d", &x[i].nilai);

puts("");

}

}

void backup(siswa a[], siswa b[], int x){

int i;

for(i=0 ; i<x ; i++){

b[i] = a[i];

}

}

void copy\_array(siswa a[], siswa b[], int x){

int i;

for(i=0 ; i<x ; i++){

a[i] = b[i];

}

}

void insertion(siswa k[])

{

int i=1, j, kondisi;

siswa key;

status = subMenu();

mode = subMenu2();

if(status==1||status==2){

while(i<n){

key=k[i];

j=i-1;

while(j>=0){

if(status==1)

{

switch(mode){

case 1: kondisi = k[j].no>key.no;

break;

case 2: kondisi = (strcmpi(k[j].nama,key.nama))>0;

break;

case 3: kondisi = k[j].nilai>key.nilai;

break;

}

}

else

{

switch(mode){

case 1: kondisi = k[j].no<key.no;

break;

case 2: kondisi = (strcmpi(k[j].nama,key.nama))<0;

break;

case 3: kondisi = k[j].nilai<key.nilai;

break;

}

}

if(kondisi){

k[j+1]=k[j];

j=j-1;

}

else

break;

}

k[j+1]=key;

i++;

}

}

tampil(k);

}

void selection(siswa k[])

{

int i=0, j, min, kondisi;

status = subMenu();

mode = subMenu2();

while(i<n-1){

min=i;

j=i+1;

while(j<n){

if(status==1)

{

switch (mode){

case 1: kondisi = k[j].no<k[min].no;

break;

case 2: kondisi = (strcmpi(k[j].nama,k[min].nama))<0;

break;

case 3: kondisi = k[j].nilai<k[min].nilai;

break;

}

}

else{

switch (mode){

case 1: kondisi = k[j].no>k[min].no;

break;

case 2: kondisi = (strcmpi(k[j].nama,k[min].nama))>0;

break;

case 3: kondisi = k[j].nilai>k[min].nilai;

break;

}

}

if(kondisi){

min=j;

}j++;

}

tukar(&k[min],&k[i]);

i++;

}

tampil(k);

}

void bubble(siswa k[], int n){

int i=0, pos\_akhir, swap=1, kondisi;

status = subMenu();

mode = subMenu2();

pos\_akhir = n-2;

while(pos\_akhir >= 0 && swap==1){

i=0;

swap=0;

while(i <= pos\_akhir){

if(status==1){

switch (mode){

case 1:

kondisi = k[i].no > k[i+1].no;

break;

case 2:

kondisi = strcmpi(k[i].nama, k[i+1].nama) > 0;

break;

case 3:

kondisi = k[i].nilai > k[i+1].nilai;

break;

default:

break;

}

}

else{

switch (mode){

case 1:

kondisi = k[i].no < k[i+1].no;

break;

case 2:

kondisi = strcmpi(k[i].nama, k[i+1].nama) < 0;

break;

case 3:

kondisi = k[i].nilai < k[i+1].nilai;

break;

default:

break;

}

}

if(kondisi){

tukar(&k[i], &k[i+1]);

swap=1;

}

i++;

}

pos\_akhir--;

}

tampil(k);

}

void shell(siswa k[], int n){

int jarak=n, swap, i=0, kondisi;

status = subMenu();

mode = subMenu2();

while(jarak>1){

jarak=jarak/2;

swap=1;

while(swap==1){

i=0;

swap=0;

while(i<(n-jarak)){

if(status==1)

kondisi = k[i].no > k[i+jarak].no;

else

kondisi = k[i].no < k[i+jarak].no;

if(kondisi){

tukar(&k[i], &k[i+jarak]);

swap=1;

}

i++;

}

}

}

tampil(k);

}

void merge(siswa data[], int l, int m, int r, int status, int mode){

int kiri1, kiri2, kanan1, kanan2, i, j, kondisi;

siswa hasil[MAKS];

kiri1=l;

kanan1=m;

kiri2=m+1;

kanan2=r;

i=l;

while((kiri1<=kanan1) && (kiri2<=kanan2)){

if(status==1){

switch(mode){

case 1: kondisi = data[kiri1].no<=data[kiri2].no;

break;

case 2: kondisi = (strcmpi(data[kiri1].nama,data[kiri2].nama))<=0;

break;

case 3: kondisi = data[kiri1].nilai<=data[kiri2].nilai;

break;

}

} else{

switch(mode){

case 1: kondisi = data[kiri1].no>data[kiri2].no;

break;

case 2: kondisi = (strcmpi(data[kiri1].nama,data[kiri2].nama))>0;

break;

case 3: kondisi = data[kiri1].nilai>data[kiri2].nilai;

break;

}

}

if(kondisi){

hasil[i]=data[kiri1];

kiri1++;

} else {

hasil[i]=data[kiri2];

kiri2++;

}

i++;

}

while(kiri1<=kanan1){

hasil[i]=data[kiri1];

kiri1++;

i++;

}

while(kiri2<=kanan2){

hasil[i]=data[kiri2];

i++;

kiri2++;

}

j=l;

while(j<=r){

data[j]=hasil[j];

j++;

}

tampil(data);

}

void mergeAwal(siswa A[], int l, int r){

status = subMenu();

mode=subMenu2();

mergeRekursif(A,l,r,status,mode);

}

void mergeRekursif(siswa A[], int l, int r, int status, int mode){

int m;

if(l<r){

m=(l+r)/2;

mergeRekursif(A,l, m,status, mode);

mergeRekursif(A,m+1,r, status, mode);

merge(A,l,m,r,status,mode);

}

}

void quick(siswa A[], int p, int r){

status = subMenu();

mode=subMenu2();

quickRekursif(A, p, r, status, mode);

}

void quickRekursif(siswa A[], int p, int r, int status, int mode){

int q; //median

if(p<r){

q=partition(A,p,r, status, mode);

quickRekursif(A, p, q, status, mode);

quickRekursif(A, q+1, r, status, mode);

}

}

int partition(siswa A[], int p, int r, int status, int mode){

int i, j;

siswa x;

x=A[p];

i=p;

j=r;

do{

if(status==1){

switch(mode){

case 1: while(A[j].no>x.no)

j--;

while(A[i].no<x.no)

i++;

break;

case 2: while((strcmpi(A[j].nama,x.nama))>0)

j--;

while((strcmpi(A[i].nama,x.nama))<0)

i++;

break;

case 3: while(A[j].nilai>x.nilai)

j--;

while(A[i].nilai<x.nilai)

i++;

break;

}

} else{

switch(mode){

case 1: while(A[j].no<x.no)

j--;

while(A[i].no>x.no)

i++;

break;

case 2: while((strcmpi(A[j].nama,x.nama))<0)

j--;

while((strcmpi(A[i].nama,x.nama))>0)

i++;

break;

case 3: while(A[j].nilai<x.nilai)

j--;

while(A[i].nilai>x.nilai)

i++;

break;

}

}

if(i<j){

tukar(&A[i],&A[j]);

j--;

i++;

} else {

return j; }

tampil(A);

}while(1);

return j;

}

void tampil(siswa a[]){

int i;

printf("No\tNama\tNilai\t\n");

puts("----------------------------------");

for (i=0;i<n;i++){

printf("%d\t%s\t%d\n",a[i].no,a[i].nama,a[i].nilai);

}

puts("");

}

void tukar(siswa \*a, siswa \*b){

siswa temp;

temp=\*b;

\*b=\*a;

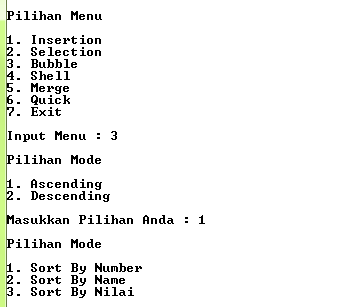
\*a=temp;

}

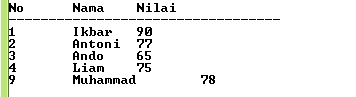
1. Capture Output



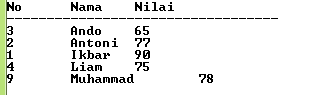
Inputan data



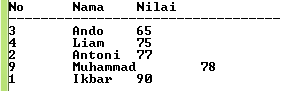
Pilihan menu



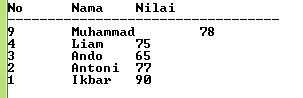
Ascending, sort by Number



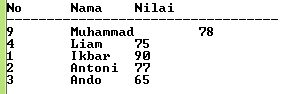
Ascending, sort by Name



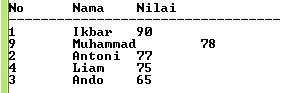
Ascending, sort by Nilai



Descending, Sort by Number



Descending, Sort by Name



Descending, Sort by Nilai

1. **Analisa**

Program diatas digunakan untuk sorting dengan tipe data integer. Dimana pilihan menu berupa iinsertion,selection,bubble,shell,merge dan quick sort.

Pertama adalah sorting dengan cara insertion. Insertion bekerja dengan cara melakukan beberapa kali iterasi, dimana setiap iterasi dilakukan pembandingan suatu bilangan dengan bilangan sebelah kirinya, jika bilangan tersebut lebih kecil maka data akan ditukar tukar, jika tidak maka bilangan yang dicek akan geser ke kanan. Hal ini terus dilakukan hingga mencapai ujung kanan dan mendapat nilai terbesar (Asc) / terkecil (Dsc) pada ujung kanan. Namun perlu diingat pada setiap iterasi berikutnya banyaknya bilangan yang dicek akan dikurangi satu dari iterasi sebelumnya.

Program sorting dengan selection bekerja dengan cara mencari nilai terkecil (Asc)/ terbesar(Dsc) dalam setiap iterasi jika ketemu maka nilai tersebut akan ditukar dengan bilangan paling kiri, jika tidak menemukan bilangan lebih kecil dari bilangan tersebut maka posisi bilangan tersebut tetap.Begitu seterusnya sampai bilangan terurut semua.

Ketiga adalah bubble short. Bubble short bekerja dengan cara membandingkan sepasang bilangan berdasarkan indexnya (i dan i+1). Jika bilangan tersebut lebih besar dari bilangan di index tertentu (i+1) maka dilakukan penukaran jika masih lebih besar dari index setelahnya maka dilakukan penukaran lagi. Jika tidak maka tidak dilakukan penukaran dan lanjutkan perbandingan pada index sepasang selanjutnya. Semua itu dilakukan sebanyak n-1. Dengan begini maka nilai terbesar pasti akan berada di paling kanan.

Keempat adalah shell sort, cara kerjanya yaitu dengan membuat sublist array lalu membandingkanya dengan jarak tertentu. Pertama - tama inisialisasikan jarak dengan n,lalu bagi jarak dengan 2. Lalu bandingkan data tersebut dengan dengan data[j+jarak] jika lebih besar(Asc)/ kecil(Dsc) maka tukar data lakukan langkah tersebut terus menerus hingga j kurang dari n- jarak, lakukan j--. Lakukan perulangan selama jarak > 1.

Kelima adalah merge short , terdapat dua fungsi untuk metode ini yaitu mergerekursi dan mergesort dimana mergerekursi dalah fungsi rekursi dengan parameter index awal dan index akhir suatu aray. Dalam mergesort array akan dibagi terus menjadi 2 bagian terus menerus hingga menjadi seoalh olah satu index lalu jadikan pasangan dengan array lainya lalu urutkan dan simpan di array penampung lakukan terus hingga array kembali utuh dan telah terurut lalu kirim nilai tersebut ke array awal.

Terakhir adalah quick sort. Pada metode ini pertama-tama partisi array menjadi dua bagian dan akan diurutkan secara rekursif. Partisi dilakukan dengan cara pilih sebuah elemen untuk menjadi pivot lalu pecah array menjadi dua bagian kemuadian lakukan increment i hingga array[i] >= pivot dan lakukan decreament j hingga a[j] <=pivot jika i<j maka tukar array[i] dengan array[j] jika tidak maka return j ke fungsi quick, lakukan langkah tersebut hingga i>=j

Untuk pada tipe data struct, sebenarnya sama saja dengan program dengan data integer, hanya saja ada tambahan untuk mengurutkannya, yaitu dengan sort by name, sort by number, dan sort by nilai. Sort by name untuk mengurutkan dari nama, sort by number untuk mengurutkan dari angka, dan sort by nilai untuk mengurutkan berdasarkan nilai.

1. **Kesimpulan**
   1. Terdapat 6 metode sorting yaitu insertion,selection,bubble,shell,merge dan quick
   2. Metode insertion bertujuan untuk menjadikan bagian sisi kiri array terurutkan sampai dengan seluruh array berhasil di urutkan.
   3. Metode selection yang dilakukan adalah pembandingan key dan penukaran elemen.
   4. Metode bubble adalah metode yang mengurutkan dengan cara membandingkan masing-masing elemen kemudian melakukan penukaran bila perlu,dengan menggelembungkan angka terbesar ke sebelah kanan.
   5. Metode shell dengan cara membandingkan suatu data dengan data lain yang memiliki jarak tertentu, sehingga membentuk sebuah sub-list kemudian dilakukan penukaran bila di perlukan
   6. Pada metode array akan dibagi menjadi dua, hal ini dilakukan terus menerus hingga masing – masing sub array hanya mempunyai satu elemen kemudian elemen – elemen tersebut dibandingkan lalu dimerge hingga kembali utuh dan dalam keadaan terurut.
   7. Untuk quick short hampir sama dengan merger sort hanya saja tidak dibagi dengan 2, melainkan menggunakan pivot.Selain itu pada quick sort juga tidak membutuhkan array sementara.