# LAPORAN

**Algoritma dan Struktur Data**

A logo with a yellow and blue design

Description automatically generated

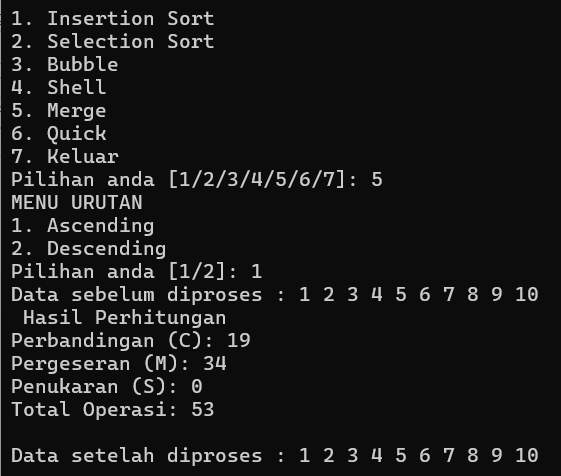
**Nama : Arif Muhammad Ihsan Marbun**

**Kelas : 1 D4 Teknik Informatika A**

**NRP : 3124600001**

1. Listing Latihan
2. Menggunakan merge and quick dengan outputan hasil dan cms
3. Program

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define n 10  int i, j, c, m, s, kondisi = 0, did\_swap = 1;  int awal[n] = {1, 2, 3, 4, 5, 6, 7, 8, 9 ,10};  int a[n];  void backup\_array();  void cek\_menu(int, int);  void tampil();  void insertion(int);  void selection(int);  void bubble(int);  void shell(int);  void mergeSort(int, int, int);  void merge(int, int, int, int);  void quickSort(int, int, int);  int partition(int, int, int);  void tukar(int \*, int \*);  void reset\_perhitungan();  void display\_perhitungan();  int main(){  int pilih1, pilih2;  while (pilih1 != 7){  reset\_perhitungan();  backup\_array();  puts("MENU METODE SORTING");  puts("1. Insertion Sort");  puts("2. Selection Sort");  puts("3. Bubble");  puts("4. Shell");  puts("5. Merge");  puts("6. Quick");  puts("7. Keluar");  printf("Pilihan anda [1/2/3/4/5/6/7]: ");  scanf("%d", &pilih1);  if (pilih1 != 7){  puts("MENU URUTAN");  puts("1. Ascending");  puts("2. Descending");  printf("Pilihan anda [1/2]: ");  scanf("%d", &pilih2);  printf("Data sebelum diproses : ");  tampil();  cek\_menu(pilih1, pilih2);  printf("Data setelah diproses : ");  tampil();  puts("");  }  }  return 0;  }  void backup\_array(){  for (i = 0; i < n; i++){  a[i] = awal[i];  }  }  void cek\_menu(int menu1, int menu2){  if (menu1 == 1){  insertion(menu2);  } else if (menu1 == 2){  selection(menu2);  } else if (menu1 == 3){  bubble(menu2);  } else if (menu1 == 4){  shell(menu2);  } else if (menu1 == 5){  mergeSort(0, n - 1, menu2);  } else if (menu1 == 6){  quickSort(0, n - 1, menu2);  } else {  exit(0);  }  display\_perhitungan();  }  void insertion(int jenis){  int key;  for (i = 1; i < n; i++){  j = i - 1;  key = a[i];  m++;  while (j >= 0){  c++;  kondisi = (jenis == 1) ? (key < a[j]) : (key > a[j]);  if (kondisi){  a[j + 1] = a[j];  m++;  j--;  } else {  break;  }  }  a[j + 1] = key;  m++;  }  }  void selection(int jenis){  int maxmin;  for (i = 0; i < n - 1; i++){  maxmin = i;  for (j = i + 1; j < n; j++){  c++;  kondisi = (jenis == 1) ? (a[j] < a[maxmin]) : (a[j] > a[maxmin]);  if (kondisi){  maxmin = j;  }  }  tukar(&a[maxmin], &a[i]);  s++;  }  }  void bubble(int jenis){  int batas = n - 1;  for (i = 0; i < n - 1; i++){  did\_swap = 0;  for (j = 0; j < batas; j++){  c++;  kondisi = (jenis == 1) ? (a[j] > a[j + 1]) : (a[j] < a[j + 1]);  if (kondisi){  tukar(&a[j], &a[j + 1]);  s++;  did\_swap = 1;  }  }  if (!did\_swap) break;  batas--;  }  }  void shell(int jenis){  int jarak = n / 2;  while (jarak >= 1){  did\_swap = 1;  while (did\_swap){  did\_swap = 0;  for (i = 0; i < n - jarak; i++){  c++;  kondisi = (jenis == 1) ? (a[i] > a[i + jarak]) : (a[i] < a[i + jarak]);  if (kondisi){  tukar(&a[i], &a[i + jarak]);  s++;  did\_swap = 1;  }  }  }  jarak /= 2;  }  }  void mergeSort(int left, int right, int jenis){  if (left < right){  int mid = (left + right) / 2;  mergeSort(left, mid, jenis);  mergeSort(mid + 1, right, jenis);  merge(left, mid, right, jenis);  }  }  void merge(int left, int mid, int right, int jenis){  int temp[n];  int i = left, j = mid + 1, k = left;  while (i <= mid && j <= right){  c++;  if ((jenis == 1 && a[i] <= a[j]) || (jenis == 2 && a[i] >= a[j])){  temp[k++] = a[i++];  m++;  } else {  temp[k++] = a[j++];  m++;  }  }  while (i <= mid){  temp[k++] = a[i++];  m++;  }  while (j <= right){  temp[k++] = a[j++];  m++;  }  for (i = left; i <= right; i++){  a[i] = temp[i];  }  }  void quickSort(int p, int r, int jenis){  if (p < r){  int q = partition(p, r, jenis);  quickSort(p, q, jenis);  quickSort(q + 1, r, jenis);  }  }  int partition(int p, int r, int jenis){  int x = a[p];  int i = p, j = r;  while (1){  while ((jenis == 1 && a[j] > x) || (jenis == 2 && a[j] < x)) {  j--;  c++;  }  while ((jenis == 1 && a[i] < x) || (jenis == 2 && a[i] > x)) {  i++;  c++;  }  if (i < j){  tukar(&a[i], &a[j]);  s++;  i++;  j--;  } else {  return j;  }  }  }  void tukar(int \*x, int \*y){  int temp = \*x;  \*x = \*y;  \*y = temp;  m += 3;  }  void tampil(){  for (i = 0; i < n; i++){  printf("%d ", a[i]);  }  puts("");  }  void reset\_perhitungan(){  c = 0;  m = 0;  s = 0;  }  void display\_perhitungan(){  printf(" Hasil Perhitungan \n");  printf("Perbandingan (C): %d\n", c);  printf("Pergeseran (M): %d\n", m);  printf("Penukaran (S): %d\n", s);  printf("Total Operasi: %d\n\n", c + m + s);  } |

 A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. Menggunakan bubble and shell tanpa outputan hasil dan cms

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  int i, j, c, m, s, kondisi = 0, did\_swap = 1;  int n;  void generate(int []);  void backup\_array(int [], int []);  void cek\_menu(int, int, int []);  void tampil(int []);  void insertion(int, int []);  void selection(int, int []);  void bubble(int, int []);  void shell(int, int []);  void mergeSort(int, int, int, int []);  void merge(int, int, int, int, int []);  void quickSort(int, int, int, int []);  int partition(int, int, int, int []);  void tukar(int \*, int \*);  void c\_m\_s();  int main(){  int pilih1, pilih2;  printf("Berapa jumlah data (maks 100000)? ");  scanf("%d", &n);  int x[n];  int a[n];  while (pilih1 != 7){  c = 0; m = 0; s = 0;  srand(time(NULL));  generate(x);  backup\_array(a, x);  puts("MENU METODE SORTING");  puts("1. Insertion Sort");  puts("2. Selection Sort");  puts("3. Bubble");  puts("4. Shell");  puts("5. Merge");  puts("6. Quick");  puts("7. Keluar");  printf("Pilihan anda [1/2/3/4/5/6/7]: ");  scanf("%d", &pilih1);  if (pilih1 != 7){  puts("MENU URUTAN");  puts("1. Ascending");  puts("2. Descending");  printf("Pilihan anda [1/2]: ");  scanf("%d", &pilih2);  cek\_menu(pilih1, pilih2, a);  c\_m\_s();  puts("");  }  }  return 0;  }  void generate(int x[]){  for (i = 0; i < n; i++){  x[i] = rand() / 1000;  }  }  void backup\_array(int a[], int x[]){  for (i = 0; i < n; i++){  a[i] = x[i];  }  }  void cek\_menu(int menu1, int menu2, int a[]){  clock\_t start, end;  double waktu;  start = clock();  if (menu1 == 1)  insertion(menu2, a);  else if (menu1 == 2)  selection(menu2, a);  else if (menu1 == 3)  bubble(menu2, a);  else if (menu1 == 4)  shell(menu2, a);  else if (menu1 == 5)  mergeSort(0, n-1, menu2, a);  else if (menu1 == 6)  quickSort(0, n-1, menu2, a);  else  exit(0);  end = clock() - start;  waktu = ((double)end) / CLOCKS\_PER\_SEC;  printf("Waktu komputasi : %g \n", waktu);  }  void insertion(int jenis, int a[]){  int key;  for (i = 1; i < n; i++){  j = i - 1;  key = a[i];  m++;  while (j >= 0){  kondisi = (jenis == 1) ? (key < a[j]) : (key > a[j]);  if (kondisi){  a[j + 1] = a[j];  m++;  c++;  j--;  } else {  break;  }  }  a[j + 1] = key;  m++;  }  }  void selection(int jenis, int a[]){  int maxmin;  for (i = 0; i < n - 1; i++){  maxmin = i;  for (j = i + 1; j < n; j++){  kondisi = (jenis == 1) ? (a[j] < a[maxmin]) : (a[j] > a[maxmin]);  if (kondisi){  maxmin = j;  }  c++;  }  tukar(&a[maxmin], &a[i]);  s++;  }  }  void bubble(int jenis, int a[]){  int batas = n - 1;  for (i = 0; i < n - 1; i++){  did\_swap = 0;  for (j = 0; j < batas; j++){  kondisi = (jenis == 1) ? (a[j] > a[j + 1]) : (a[j] < a[j + 1]);  if (kondisi){  tukar(&a[j], &a[j + 1]);  s++;  did\_swap = 1;  }  c++;  }  if (!did\_swap) break;  batas--;  }  }  void shell(int jenis, int a[]){  int jarak = n / 2;  while (jarak >= 1){  did\_swap = 1;  while (did\_swap){  did\_swap = 0;  for (i = 0; i < n - jarak; i++){  kondisi = (jenis == 1) ? (a[i] > a[i + jarak]) : (a[i] < a[i + jarak]);  if (kondisi){  tukar(&a[i], &a[i + jarak]);  s++;  did\_swap = 1;  }  c++;  }  }  jarak /= 2;  }  }  void mergeSort(int left, int right, int jenis, int a[]){  if (left < right){  int mid = (left + right) / 2;  mergeSort(left, mid, jenis, a);  mergeSort(mid + 1, right, jenis, a);  merge(left, mid, right, jenis, a);  }  }  void merge(int left, int mid, int right, int jenis, int a[]){  int temp[n];  int i = left, j = mid + 1, k = left;  while (i <= mid && j <= right){  c++;  if ((jenis == 1 && a[i] <= a[j]) || (jenis == 2 && a[i] >= a[j])){  temp[k++] = a[i++];  m++;  } else {  temp[k++] = a[j++];  m++;  }  }  while (i <= mid){  temp[k++] = a[i++];  m++;  }  while (j <= right){  temp[k++] = a[j++];  m++;  }  for (i = left; i <= right; i++){  a[i] = temp[i];  }  }  void quickSort(int p, int r, int jenis, int a[]){  if (p < r){  int q = partition(p, r, jenis, a);  quickSort(p, q, jenis, a);  quickSort(q + 1, r, jenis, a);  }  }  int partition(int p, int r, int jenis, int a[]){  int x = a[p];  int i = p, j = r;  while (1){  while ((jenis == 1 && a[j] > x) || (jenis == 2 && a[j] < x)) {  j--;  c++;  }  while ((jenis == 1 && a[i] < x) || (jenis == 2 && a[i] > x)) {  i++;  c++;  }  if (i < j){  tukar(&a[i], &a[j]);  s++;  i++;  j--;  } else {  return j;  }  }  }  void tukar(int \*x, int \*y){  int temp = \*x;  \*x = \*y;  \*y = temp;  m += 3;  }  void c\_m\_s(){  printf("Hasil compare : %d\n", c);  printf("Hasil movement : %d\n", m);  printf("Hasil swap : %d\n", s);  } |

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.