LAPORAN
ALGORITMA DAN PEMROGRAMAN
INSERTION SORT

Oleh:

M. Andi Divangga Pratama	1203210005
Qori Emalia Putri Mafidah	1203210024
Elma Saskia Mawardi	1203210056
M. Ali Akbar Velayati P. A.	1203210072
M. Hanafi Choirullah	1203210076
Al Farel Ilyas Hariyanto	1203210088
Nurul Azizi Hasibuan	1203210090
Yudhistira Ar Rahmanu S.	1203210096

Kelompok 4

IT TELKOM SURABAYA
FAKULTAS TEKNOLOGI INFORMASI DAN BISNIS
PROGRAM STUDI S1 INFORMATIKA
TAHUN AKADEMIK 2021/2022

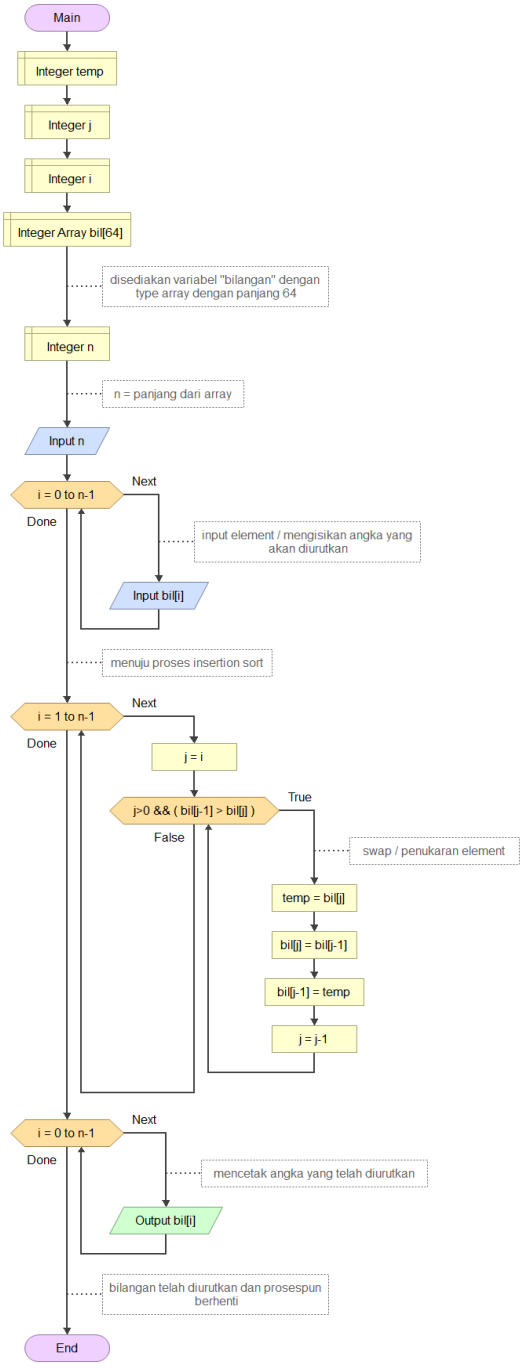
DAFTAR ISI

ISI

1. Pengertian *Insertion Sort*

Insertion Sort adalah algoritma sederhana untuk mengurutkan sekumpulan data dengan cara membandingkan semua data dimulai dari dua data pertama dan dilanjutkan hingga data ke-n. *Insertion sort* sama halnya seperti permainan kartu *bridge* (Kartu remi). Algoritma *Insertion Sort* dapat mengurutkan data secara *ascending* (Pengurutan dari data terkecil hingga terbesar) atau *descending* (Pengurutan data dari terbesar hingga terkecil).

2. *Flowchart*



3. *Pseudocode*

```
Function Main
   Declare Integer temp
   Declare Integer j
   Declare Integer i
   Declare Integer Array bil[64]

   ... disediakan variabel "bilangan" dengan type array dengan panjang 64
   Declare Integer n

   ... n = panjang dari array
   Input n
   For i = 0 to n-1

      ... input element / mengisikan angka yang akan diurutkan
      Input bil[i]
   End

   ... menuju proses insertion sort
   For i = 1 to n-1
      Assign j = i
      While j>0 && ( bil[j-1] > bil[j] )

         ... swap / penukaran element
         Assign temp = bil[j]
         Assign bil[j] = bil[j-1]
         Assign bil[j-1] = temp
         Assign j = j-1
      End
   End
   For i = 0 to n-1

      ... mencetak angka yang telah diurutkan
      Output bil[i]
   End

   ... bilangan telah diurutkan dan proses pun berhenti
End
```

4. Kode Program

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int n, i, j, temp;
6      int arr[64];
7
8      //JUDUL
9      printf("----------INSERTION SORT----------\n\n");
10
11     //INPUT BANYAKNYA BILANGAN YANG INGIN DIURUTKAN
12     printf("Masukkan Jumlah Angka : ");
13     scanf("%d", &n);
14
15     puts(" ");
16
```

- **Baris 1 –** Cantumkan *header*, yaitu stdio.h
- **Baris 3 –** Tulis tipe data berupa *integer* untuk *main*
- **Baris 5 –** Deklarasikan variabel, yaitu n untuk menyimpan banyaknya bilangan, i untuk *looping for*, j untuk *looping while*, dan temp untuk *temporary* atau tempat kosong yang berguna sebagai tempat untuk menyimpan sesuatu
- **Baris 6 –** Deklarasikan variabel, yaitu arr dengan nilai *array* sebanyak 64 yang berfungsi untuk menyimpan setiap elemen/ bilangan
- **Baris 9 –** Tulis fungsi *printf* untuk menampilkan *output* berupa judul
- **Baris 12 –** Tulis fungsi *printf* untuk menampilkan *output* berupa perintah agar pengguna memasukkan jumlah angka yang diinginkan
- **Baris 13 –** Tulis fungsi *scanf* untuk mengambil *input* jumlah angka dari pengguna
- **Baris 15 –** Tulis fungsi *puts* untuk memberi jarak dengan *line* baru

```
17     //INPUT ANGKA YANG INGIN DIURUTKAN
18     printf("---------MASUKKAN %d ANGKA---------\n\n", n);
19     for (i = 0; i < n; i++)
20     {
21         printf("Masukkan Angka Ke-%d : ",i+1);
22         scanf("%d", &arr[i]);
23     }
24
```

- **Baris 18 –** Tulis fungsi *printf* untuk menampilkan *output* berupa perintah agar pengguna memasukkan angka yang ingin diurutkan sesuai dengan jumlah yang di*input*kan pada baris 12 dan 13
- **Baris 19 –** Gunakan fungsi perulangan *for*, nantinya angka yang di*input*kan satu persatu akan disimpan pada arr[0] hingga arr[n]

```
25      //PROSES MENGURUTKAN
26      for (i = 1 ; i <= n-1 ; i++)
27      {
28          j = i;
29              while ( j > 0 && arr[j-1] > arr[j])
30              {
31                  //MELAKUKAN SWAP ANGKA JIKA TIDAK URUT
32                  temp     = arr[j];
33                  arr[j]   = arr[j-1];
34                  arr[j-1] = temp;
35                  j--;
36              }
37      }
38
39      puts(" ");
40
```

- **Baris 26 –** Gunakan fungsi perulangan *for* untuk melakukan perpindahan mulai dari pencocokan, di mana mulai darinya akan dimulai pada *index* ke-1 atau data ke-2. Mulai dari atau *index* awal pencocokan akan terus bertambah hingga *index* ke-n
- **Baris 29 –** Gunakan fungsi perulangan *while* untuk mengurutkan data, di sini nilai variabel yang dimasukkan akan diperiksa apakah memenuhi kondisi atau tidak. Jika sesuai, maka *while* akan men-*swap* atau menukar *index* yang dicocokkan satu sama lain hingga kondisi sudah tidak memenuhi. Jika tidak memenuhi, maka proses *while* selesai dan dilanjutkan ke *looping for*
- **Baris 39 -** Tulis fungsi *puts* untuk memberi jarak dengan *line* baru

```
41      //MENAMPILKAN ANGKA SETELAH DIURUTKAN
42      printf("---------SETELAH DIURUTKAN---------\n");
43
44      puts(" ");
45
46      for (i = 0; i <= n - 1; i++)
47      {
48          printf("%d, ", arr[i]);
49      }
50      return 0;
51  }
52
```

- **Baris 42 –** Tulis fungsi *printf* untuk menampilkan output pada layar
- **Baris 44 –** Tulis fungsi *puts* untuk memberi jarak dengan *line* baru
- **Baris 46 –** Gunakan fungsi perulangan *for* untuk menampilkan angka setelah diurutkan

5. Hasil *Run* Program

```
----------INSERTION SORT------------

Masukkan Jumlah Angka : 6

----------MASUKKAN 6 ANGKA----------

Masukkan Angka Ke-1 : 8
Masukkan Angka Ke-2 : 3
Masukkan Angka Ke-3 : 5
Masukkan Angka Ke-4 : 2
Masukkan Angka Ke-5 : 9
Masukkan Angka Ke-6 : 1

----------SETELAH DIURUTKAN---------

1
2
3
5
8
9

Process returned 0 (0x0)   execution time : 15.259 s
Press any key to continue.
```

KESIMPULAN

Pada intinya, *Insertion Sort* sama seperti mengurutkan kartu remi/ *brigde*, di mana kita meng-*insert* atau memasukkan kartu di antara kartu-kartu lain sesuai urutan. Perbedaannya hanyalah pada proses pelaksanaan atau eksekusi, di mana komputer meng-*insert* (memasukkan) dan mengecek secara runtun hingga menemukan tempat yang sesuai dengan urutan. Oleh karena itu, dapat kita ketahui bahwa *Insertion Sort* mempunyai beberapa kelebihan dan kelemahan. Kelebihan *Insertion Sort* adalah ketika banyaknya data yang diolah relatif sedikit maka metode *sorting* ini akan sangat efektif dibandingkan dengan metode *sorting* lainnya. Namun sebaliknya, kelemahannya adalah ketika data yang diolah relatif banyak membuat metode ini kurang efektif karena jalannya program yang meng-*insert* (memasukkan) lalu membandingkan satu persatu data hingga sesuai urutannya.

DAFTAR PUSTAKA

Bhojasia, Manish. (Tanpa Tahun). *C Program to Implement Insertion Sort.* Diakses pada tanggal 14 Desember 2021. https://www.sanfoundry.com/c-program-insertion-sort/.

Gozali, W., & Aji, A. F. *Pemrograman Kompetitif Dasar Panduan Memulai OSN Informatika, ACM-ICPC, dan sederajat.* 1.9. Edited by I. W. Kurnia & S. Effend. CV Nulisbuku Jendela Dunia.

Tanzil, Fidelson. (2019). *Insertion Sort*. Diakses pada tanggal 19 Desember 2021. https://socs.binus.ac.id/2019/12/30/insertion-sort/.