**LAPORAN UAS ALGORITMA DAN PEMROGRAMAN LANJUT**

**LAPORAN UAS**

Diajukan Untuk Memenuhi Tugas Akhir Semester

Algoritma dan Pemrograman Lanjut

|  |
| --- |
| **Konsep Penjelasan :** |
| Bubble Sort |
| Insertion sort |
| Selection sort |
| *Priority* Queue |
| Heap sort |
| Merge sort |
| Algoritma Faktorial dengan Rekursif |

**Disusun Oleh :**

|  |  |
| --- | --- |
| **Leni Salsabila Apriliyani** | **180914047** |
| **3IF-01** | |

****

**STMIK & POLITEKNIK K O M P U T E R N I A G A L P K I A**

**PROGRAM STUDI MANAJEMEN INFORMATIKA**

**KONSENTRASI TEKNIK INFORMATIKA**

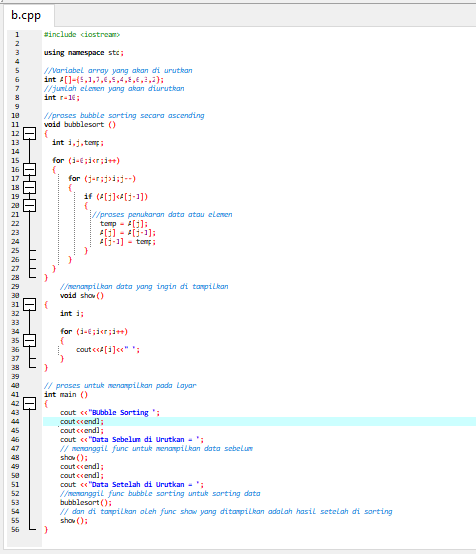
**BANDUNG**

**2021**

1. **Bubble Sort**

*Bubble sort* adalah membandingkan elemen pertama dan elemen yang berikut lalu di *swap* jika data atau elemen sesuai dengan ketentuan yang ditentukan (kecil ke besar).

* **Coding Program**

****

* **Penjelasan Coding**

**Baris 6** = Membuat variable array yang akan di eksekusi atau diurutkan

**Baris 8**  = Menentukan jumlah elemen yang akan di eksekusi

**Baris 11 – 28** = Proses *bubble sorting* secara *ascending*

**Baris 11** = *function*  untuk *bubble sorting*

**Baris 13** = membuat variable

**Baris 22 – 24** = proses penukaran data atau elemen

**Baris 30** = membuat *function*  untuk menampilkan data

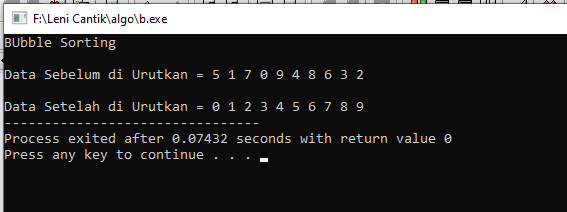
**Baris 30-38** = proses *function*  untuk menampilkan data

**Baris 40-56** = proses untuk menampilkan data dilayar

**Baris 47** = memanggil *function*  untuk menampilkan data yang belum di *sorting*

Baris 53 dan 55 = memampilkan *function*  proses *bubble sorting* dan tampilan *function*  show yang telah di *sorting*.

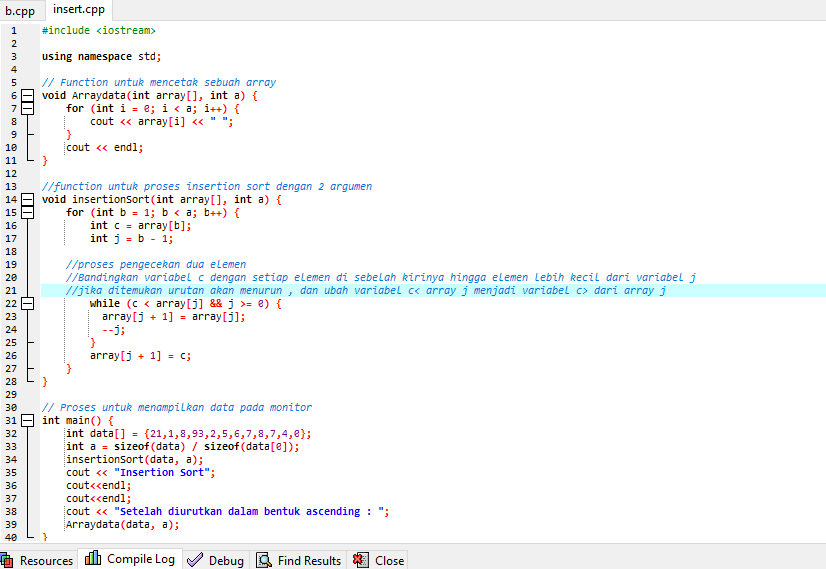
* **Sreenshoot Running**

****

1. **Insertion Sort**

*Insertion sort* adalah pengurutan data dengan cara menyisipkan elemen data pada posisi yang tepat. Jadi di cek satu per satu dari awal sampai akhir dan di *swap*.

* **Coding Program**

****

* **Penjelasan**

**Baris 6 – 11 =** proses sebuah *function*  untuk mencetak sebuah array

**Baris 14 – 28 =** sebuah *function*  untuk proses sorting insertion sort

**Baris 14 – 17 =** pada *for* untuk *looping* elemen keseluruhan dan int c untuk m

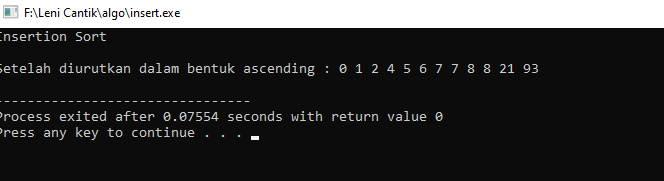
**Baris 22 -27 =** Proses mengecekan 2 elemen dimana ada perbandingan variable c dengan setiap elemen di sebelah kirinya hingga elemen lebih kecil dari variable j jika ditemukan urutan akan menurun , dan ubah variabel c< array j menjadi variabel c> dari array j.

**Baris 32 =** menentukan isi dari sebuah array atau isi elemen

**Baris 33 =** untuk mengukur panjang sebuah array

**Baris 34 =** *function InsertionSort* yang memiliki 2 argumen yang pertama data (yaitu data yang akan muncul atau di urutkan atau juga isi dari sebuah elemen) dan argument ke-2 yaitu (variable untuk mengukur panjang sebuah array yang dimasukan)

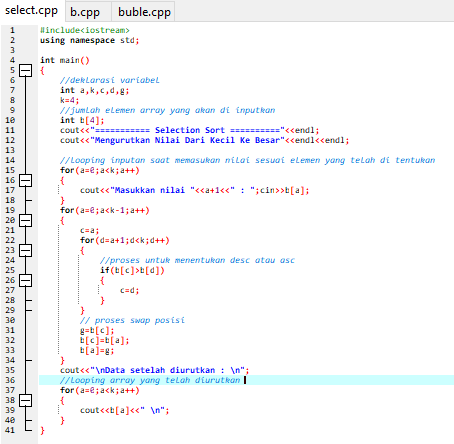
* **Screenshoot Running**

****

1. **Selection Sort**

Selection Sort adalah metode sorting dimana elemen- elemen di perbandingkan satu-persatu sampai pada elemen terakhir dan disusun berdasarkan ketentuan (kecil ke besar ).

* **Coding Program**

****

* **Penjelasan**

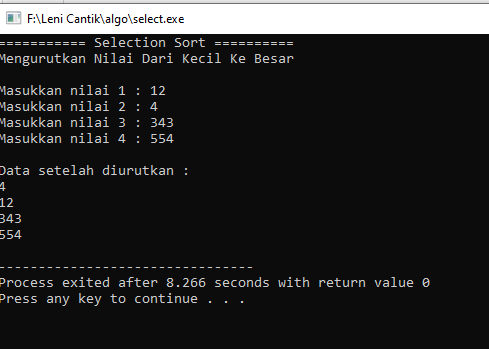
**Baris 7-10** = Pendeklarasian variabel dan array yang akan digunakan dalam program

**Baris 15-18** = Proses inputan yang disimpan dalam array yang dilakukan dalam perulangan

**Baris 19-22** = Proses pengurutan antara elemen satu dengan yang lain dan apabila elemen satu lebih kecil daripada elemen berikutnya (mengurtkan kecil ke besar) maka proses pertukaran akan terjadi pada pada baris 31-33.

**Baris 37-40** = Setelah pengurutan berhasil maka nilai akan dicetak/ditampilkan pada baris ini.

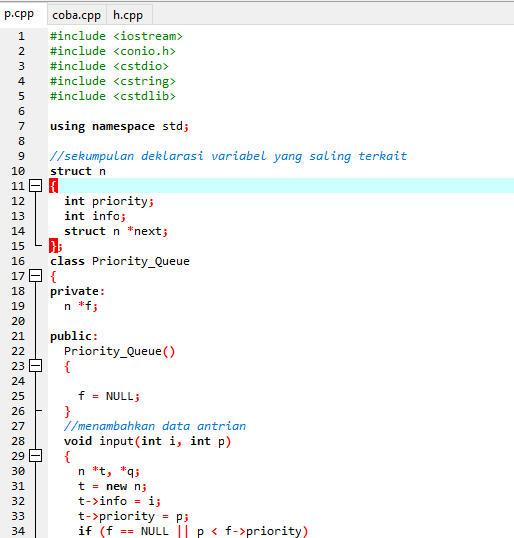
* **Screenshoot Running**

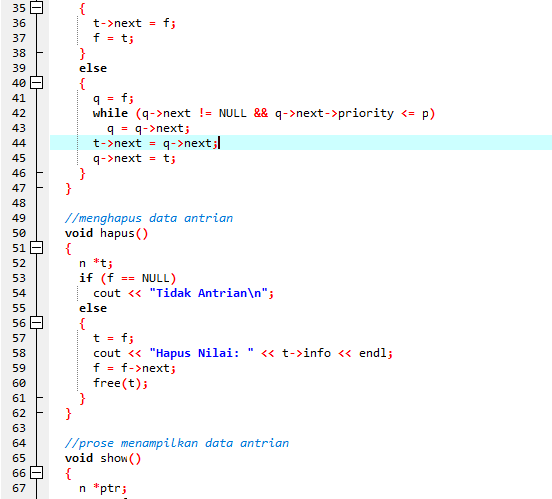
****

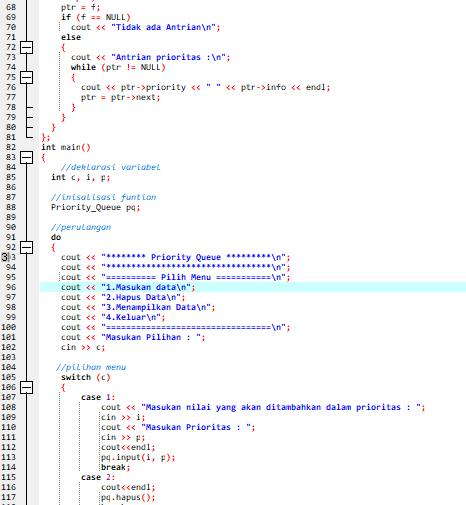
1. ***Priority* Queue**

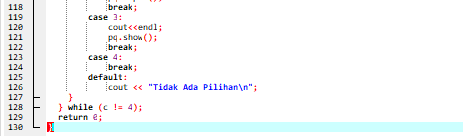
*Priority queue* adalah kumpulan semua elemen yang mempunyai urutan perbandingan (*priority*).

* **Coding Program**

****

****

****

****

* **Penjelasan**

**Baris 10-15 =** struct yaitu sekumpula deklarasi variabel yang saling terkait

**Baris 16-81 =** kelas *priority* untuk memproses setiap *function*  tertentu

**Brais 28-47 =** *function*  yang memproses untuk menambahkan data antrian

**Baris 50-62 =** *function*  yang memproses hapusnya data antrian atau nilai yang terakhir diinputkan

**Baris 53 =** jika variabel f atau data yang di masukan tidak ada maka akan di jalankan baris 54

**Baris 57 =** jika ada data yang akan di hapus ada maka akan dijalankan baris 58-61

**Baris 65-80 =** *function*  yang menampilkan data antrian

**Baris 69 =** jika data antrian belum diinputkan sama sekali maka akan dijalankan baris 70

**Baris 71-80 =** jika data antrian tersedia maka baris ini akan ditampilkan

**Baris 85 =** deklarasi variabel

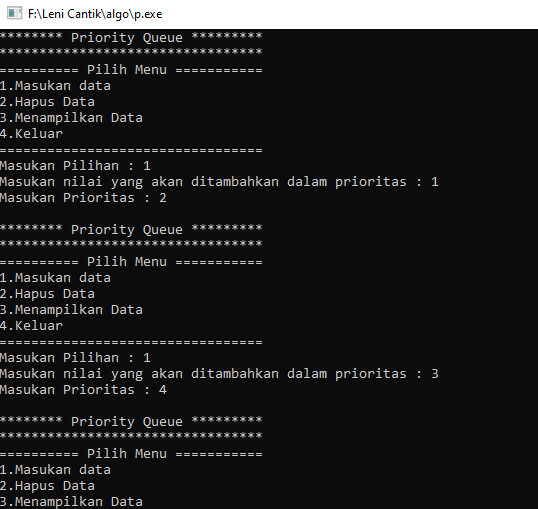
**Baris 88 =** inisialisasi *function*

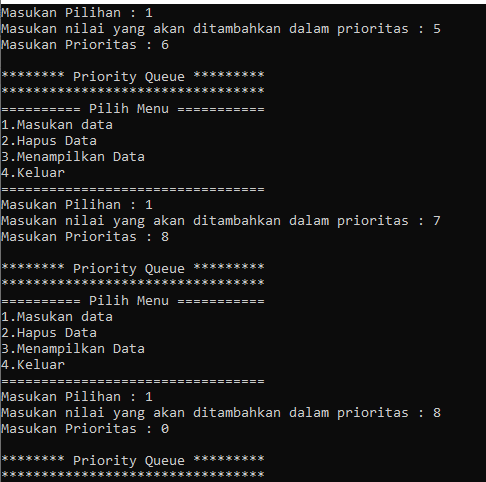
**Baris 91-128 =** perulangan jika user memilih menu 1-3

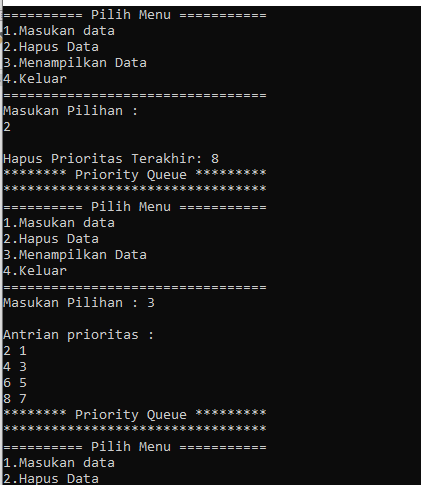
**Baris 105-127 =** pilihan menu

**Baris 128 =** jika user input menu no 4 maka akan di hentikan atau dijalankan baris 129

* **Screenshoot Running**

****

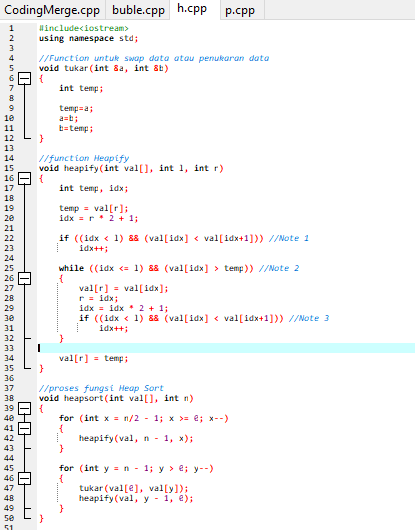
****

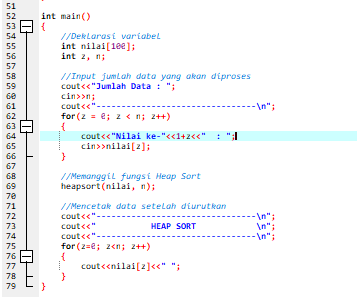
****

1. **Heap Sort**

*Heap sort* adalah sebuah metode *sorting* (pengurutan) angka pada sebuah array dengan cara menyerupai *binary tree*, yaitu dengan cara memvisualisasikan sebuah array menjadi sebuah *binary tree* yang nantinya pada *binary tree* tersebut nilai pada masing-masing index array akan diurutkan.

* **Coding Program**

****

****

* **Penjelasan**

**Baris 5 -12 =** *Function* untuk *swap* data atau penukaran data

**Baris 15-35 =** *Function heapify*

**Baris 22-23 =** Pengurutan secara ascending menggunakan tanda “<”, sedangkan pengurutan secara descending menggunakan tanda “>”

**Baris 25-29 =** Pengurutan secara ascending menggunakan tanda “>”, sedangkan pengurutan secara descending menggunakan tanda “<”

**Baris 30-32 =** Pengurutan secara ascending menggunakan tanda “<”, sedangkan pengurutan secara descending menggunakan tanda “>”

**Baris 38-50 =** Proses *function heap sort*

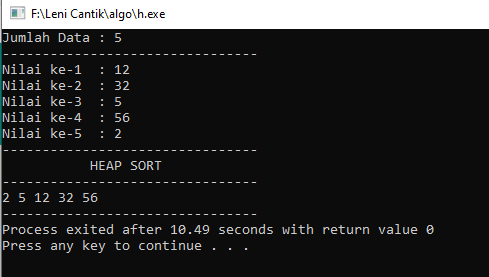
**Baris 55-56 =** Deklarasi variabel

**Baris 59-66 =** *Input* jumlah data yang akan diproses

**Baris 69 =** Memanggil fungsi heapsort dengan 2 argumen

**Baris 72-79 =** Mencetak data yang telah diurutkan

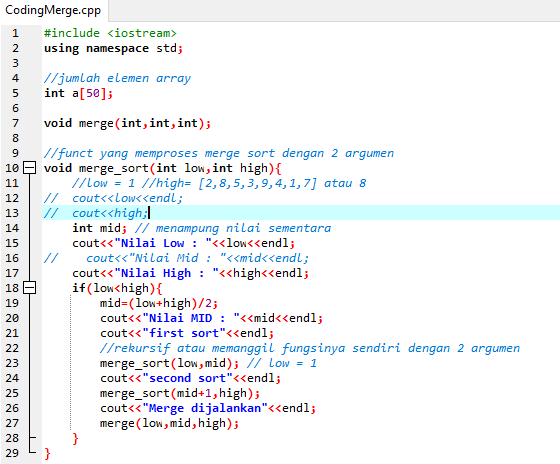
* **Screenshoot Running**

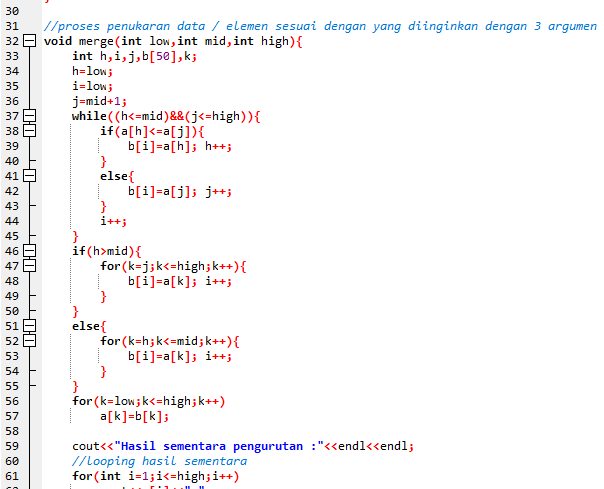
****

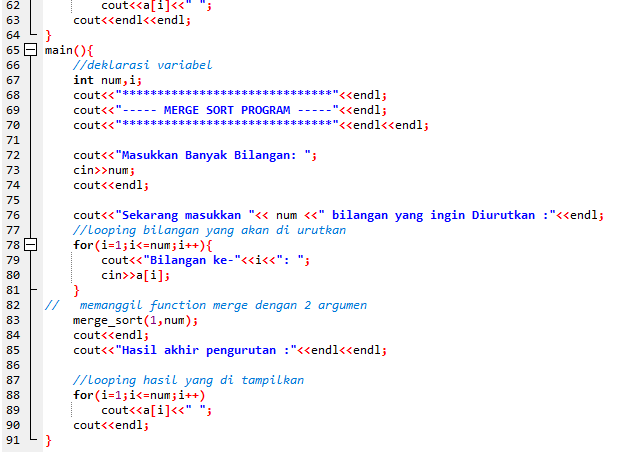
1. **Merge sort**

Merge Sort adalah metode pengurutan data yang dilakukan dengan cara divide and conquer yaitu data dibagi menjadi sub kumpulan – sub kumpulan yang kemudian sub kumpulan tersebut diurutkan secara terpisah, dan kemudian digabungkan kembali dengan metode merging. Metode Merge Sort merupakan metode yang menggunakan fungsi rekursi untuk penyelesaiannya.

* **Coding Program**

****

****

****

* **Penjelasan**

**Baris 5 =** deklarasi variable array dengan menentukan jumlah elemen array

**Bais 6 =** *function* dengan 3 argumen

**Baris 10 =** *function* yang memproses merge sort dengan 2 argumen

**Baris 14 =** variabel yang menampung nilai sementara

**Baris 18 =** jika low kurang dari nilai high maka proses akan dijalankan

**Baris 19 =** untuk menentukan nilai sementara yaitu nilai low + nilai high dan dibagi nilai dibagi 2

**Baris 23 =** rekursif atau memanggil fungsinya sendiri dengan 2 argumen

**Baris 32 =** proses penukaran data / elemen yang sesuai dengan yang diingikan dengan cara menggunakan 3 argumen

**Baris 61 =** *looping* hasil sementara

**Baris 65 – 91 =** *function* untuk menampilkan dilayar monitor

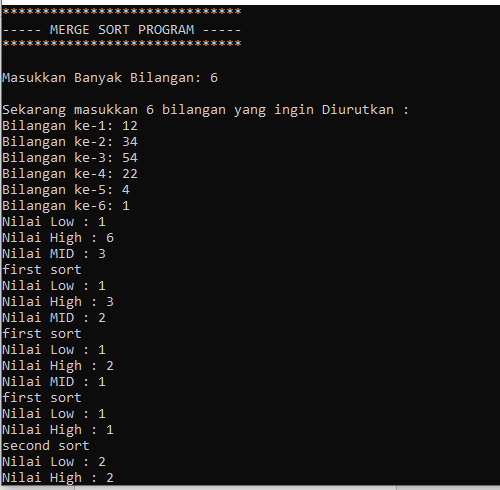
**Baris 67 =** deklarasi variabel untuk elemen

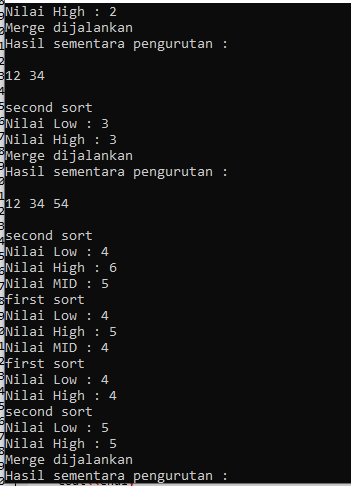
**Baris 78 =** *looping* bilangan yang akan diurutkan

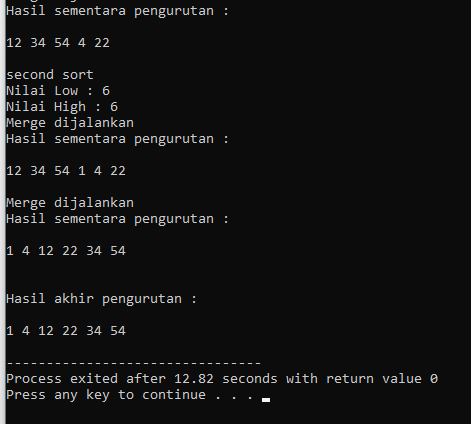
**Baris 83 =** memanggil *function merge* dengan 2 argumen

**Baris 88 =** *looping* hasil yang akan ditampilkan

* **Screenshoot Running**

****

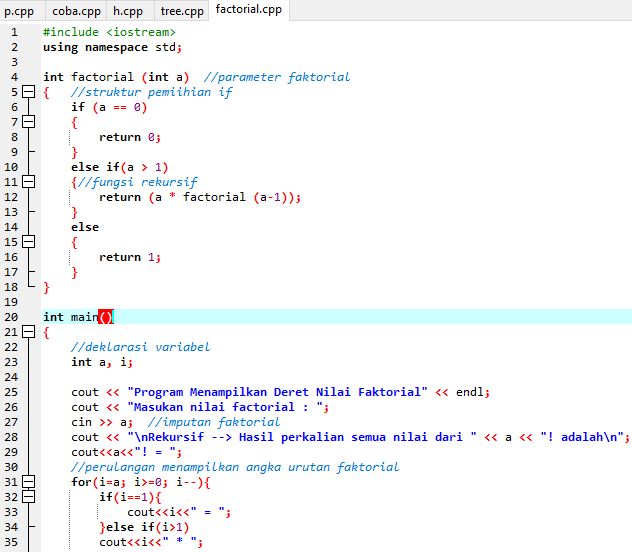
****

****

1. **Algoritma Factorial dengan rekursif**

**Faktorial** adalah sebuah fungsi permutasi dari bilangan bulat positif yang menggunakan notasi n! . **Faktorial** akan mengalikan bilangan dengan bilangan berikutnya hingga terakhir dikalikan angka 1. **Rekursif** adalah suatu proses dari sebuah fungsi yang dapat memanggil dirinya sendiri secara berulang-ulang. Maka dari itu *factorial* dengan rekursif ialah fungsi permutasi bilangan bulat dengan menggunakan proses fungsi rekursif atau memanggi dirinya sendiri.

* **Coding Program**

****

* **Penjelasan**

**Baris 4 =** parameter factorial

**Baris 6-17 =** struktur pemilihan *if*

**Baris 12 =** fungsi rekursif atau memanggil dirinya sendiri

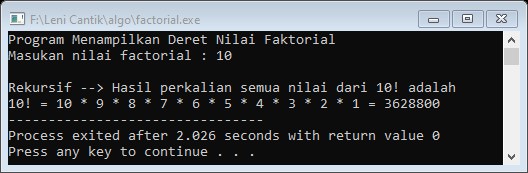
**Baris 23 =** deklarasi variabel

**Baris 27 =** inputan faktorial

**Baris 31-36 =** perulangan menampilkan angka urutan *factorial*, jika i==1 maka akan menampilkan sama dengan (=), jika i lebih dari 1 maka akan menampilkan urutan angka factorial dan di ikuti dengan (\*)

**Baris 37 =** pemanggilan dan menampilkan faktorial

* **Screenshoot Running**

****