**LAPORAN TUGAS STRUKTUR DATA**

**KASUS PERMASALAHAN DUNIA NYATA YANG DAPAT DISELESAIKAN DENGAN MENGGUNAKKAN FUNGSI REKURSIF**

****

Dosen Pembimbing:

Dr. Ricky Eka Putra, S.Kom., M.Kom.

Disusun oleh:

Cornelius Louis Nathan 23051204085

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**UNIVERSITAS NEGERI SURABAYA**

**2024**

# KATA PENGANTAR

Puji syukur saya haturkan kepada Tuhan Yang Maha Esa sehingga atas rahmat-Nya. Saya dapat menyelesaikan laporan tugas struktur data yang berjudul “Kasus Permasalahan Dunia Nyata Yang Dapat Diselesaikan Dengan Menggunakkan Rekursif” sebagai tugas struktur data tepat pada waktunya.

Adapun tujuan penulisan laporan ini adalah untuk mengetahui apa itu fungsi rekursif dan penggunaan fungsi rekursif dalam kasus permasalahan dunia nyata. Pembuatan laporan ini sekaligus menjadi tugas yang mengisi nilai dari Mata Kuliah Struktur Data.

Penyusunan laporan ini tidak lepas dari bantuan dan dukungan. Sehingga, kami menyampaikan terima kasih kepada pihak yang telah mendukung pembuatan laporan ini. Terutama kepada Bapak Ricky Eka Putra, S.Kom., M.Kom., selaku Dosen Pembimbing Mata Kuliah Struktur Data, atas petunjuk, didikan, dan arahan yang sangat membantu dalam penyusunan laporan ini.

Kami menyadari bahwasanya penulisan laporan ini jauh dari kata sempurna, sehingga kami menerima saran dan kritik yang membangun dari pembaca guna untuk perbaikan dan perkembangan di masa mendatang. Semoga penyusunan laporan ini juga berguna bagi para pembaca dalam memahami apa itu fungsi rekursif dan bagaimana penggunaan fungsi rekursif dalam kasus permasalahan dunia nyata.

Surabaya, 04 April 2024

Penulis

# DAFTAR ISI

[KATA PENGANTAR ii](#_Toc163735750)

[DAFTAR ISI 3](#_Toc163735751)

[BAB I PENDAHULUAN 4](#_Toc163735753)

[1.1 Latar Belakang 5](#_Toc163735754)

[1.2 Rumusan Masalah 5](#_Toc163735755)

[1.3 Tujuan 5](#_Toc163735756)

[BAB II KAJIAN PUSTAKA 6](#_Toc163735758)

[2.1 Fungsi Rekursif 6](#_Toc163735759)

[2.2 Pengimplementasian Fungsi Rekursif terhadap Kasus Permasalahan di Dunia Nyata 12](#_Toc163735760)

[2.3 Perbandingan antara Fungsi Biasa dengan Fungsi Rekursif 19](#_Toc163735761)

[2.4 Output yang dihasilkan dari Fungsi Rekursif 21](#_Toc163735762)

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Penggunaan fungsi adalah salah satu bagian fundamental dalam pemrograman komputer. Selain mendukung seorang *programmer* untuk menerapkan prinsip *DRY* (*Don’t Repeat Yourself*), fungsi digunakan untuk memudahkan pembagian program menjadi bagian-bagian yang lebih kecil, sehingga program lebih mudah dikelola dan dimodifikasi.

Fungsi memiliki tugas yang disesuaikan dalam suatu program dan dapat dipanggil dari bagian lain dalam program. Asalkan fungsi telah dideklarasikan dan dibuat dengan logika yang benar, fungsi dapat mengambil argumen sebagai input, menjalankan instruksi yang diperintahkan dan mengembalikan hasil sebagai output.

Dalam contoh kasus permasalahan di dunia nyata, andaikan terdapat suatu restoran menginginkan program yang bisa menambah data reservasi, menampilkan daftar reservasi, dan mencari nama reservasi. Sehingga, setidaknya membutuhkan tiga fungsi untuk melaksanakan instruksi. Tetapi, pembuatan fungsi biasa akan memicu adanya redundansi variable dan data. Sehingga, untuk mengatasi hal tersebut, fungsi-fungsi tersebut akan dirubah menjadi fungsi rekursif.

## 1.2 Rumusan Masalah

1. Apa pengertian dari fungsi rekursif?
2. Bagaimana pengimplementasian fungsi rekursif terhadap kasus permasalahan di dunia nyata?
3. Bagaimana perbandingan antara program dengan fungsi biasa dengan program dengan fungsi rekursif?
4. Bagaimana output yang dihasilkan dari penggunaan program fungsi rekursif?

## 1.3 Tujuan

1. Memahami pengertian dari fungsi rekursif.
2. Memahami pengimplementasian fungsi rekursif terhadap kasus permasalahan di dunia nyata.
3. Memahami perbandingan antara program dengan fungsi biasa dengan program dengan fungsi rekursif.
4. Memahami output yang dihasilkan dari penggunaan program fungsi rekursif

# BAB II

# KAJIAN PUSTAKA

## 2.1 Fungsi Rekursif

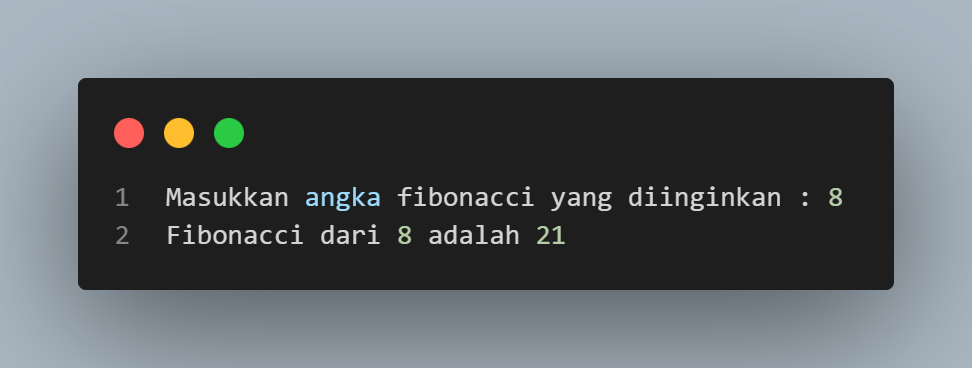
Fungsi rekursif adalah fungsi yang berisi aksi untuk memanggil dirinya sendiri. Fungsi rekursif terus memanggil dirinya sendiri sampai suatu kondisi terpenuhi. Rekursif sebenarnya memiliki kesamaan dengan iterasi, dimana kita perlu memastikan agar algoritma tidak mengalami *infinite loop*. Contoh perumpamaan dalam kehidupan sehari-hari adalah seorang ibu yang memotong sayur sampai menjadi bagian kecil. Maka ibu itu akan terus memotong sampai potongan sayur menjadi terlalu kecil atau sampai tidak bisa dipotong lagi.

Rekursif terbagi menjadi tiga bagian, yaitu pemecahan masalah, pemanggilan fungsi, dan *base case*. Pemecahan masalah dalam rekursif adalah memecah atau membagi masalah menjadi bagian yang lebih kecil. Perlu dipastikan pula setiap masalah dapat diselesaikan dengan cara yang sama. Pemanggilan fungsi adalah ciri khas dari fungsi rekursif. Fungsi rekursif perlu memanggil fungsi itu sendiri agar mencapai kondisi yang diinginkan. *Base case* adalah kondisi dimana fungsi berhenti memanggil dirinya sendiri. *Base case* lah yang menghentikan proses rekursi dan mengembalikan output sesuai ekspetasi.

Contoh penggunaan fungsi rekursif secara sederhana adalah penghitungan deret Fibonacci, yang memiliki rumus F(n-1) + F(n-2). Rumus F(n-1) + F(n-2) adalah contoh dari pemecahan masalah, dan apabila diperhatikan dari deret berikut, yaitu 0,1,1,2,3,5,8,… . Maka angka 0 dan 1 adalah *base case* dari deret Fibonacci. Sehingga, apabila penghitungan deret Fibonacci dibuat menjadi fungsi rekursif.



Maka, output yang dihasilkan adalah



    if(N == 0){

        return 0;

    }

    if(N == 1){

        return 1;

    }

N == 0 dan N == 1 adalah *base case*. Apabila N telah mencapai 0 atau 1, maka akan mengembalikan nilai 0 atau 1.

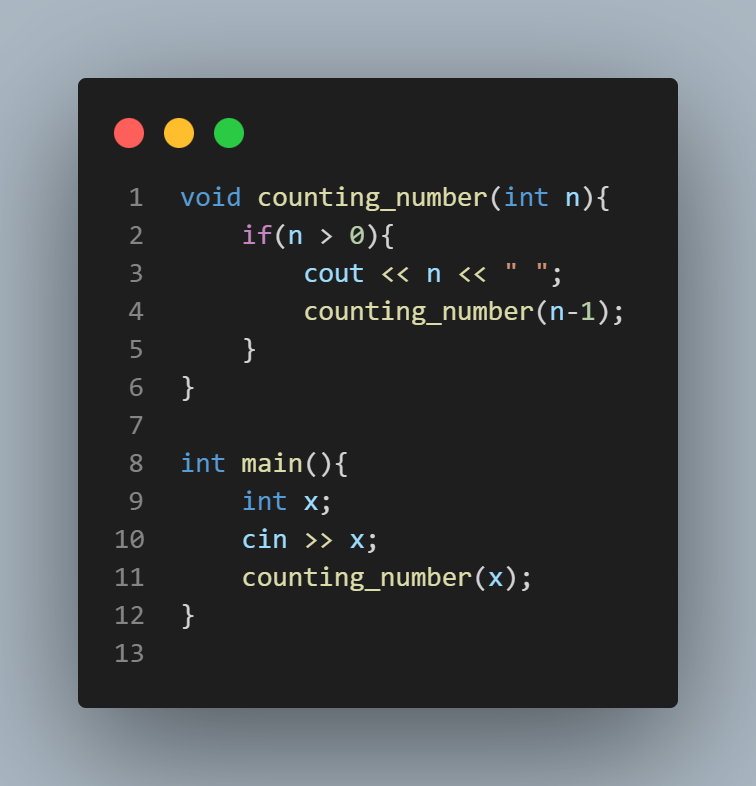
    else{

        return Fibonacci(N-1) + Fibonacci(N-2);

    }

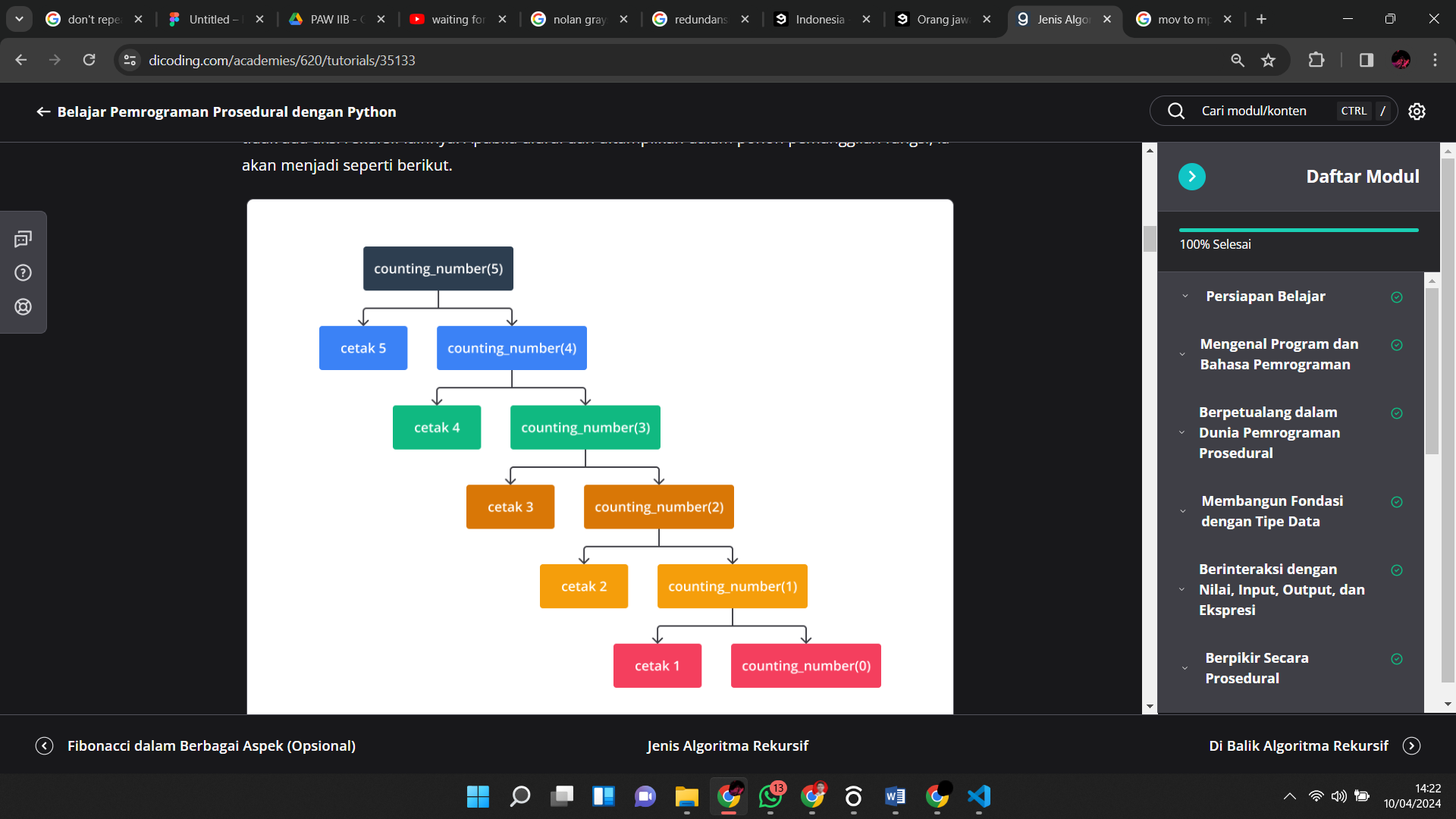
Jika belum mencapai angka 0 atau 1,maka dilakukanlah proses pemanggilan fungsi dengan mengurangi N untuk mencapai angka 1 dan 0. Sehingga disitulah terjadi *looping* Fibonacci.

Fungsi rekursif juga dibedakan menurut jenisnya. Berdasarkan struktur pemanggilan fungsi, terdapat *linear recursion* dan *tree recursion*. *Linear recursion* adalah fungsi rekursif yang hanya memanggil sekali fungsi itu sendiri dalam setiap pemanggilannya. Sebagai contoh fungsi counting\_number();



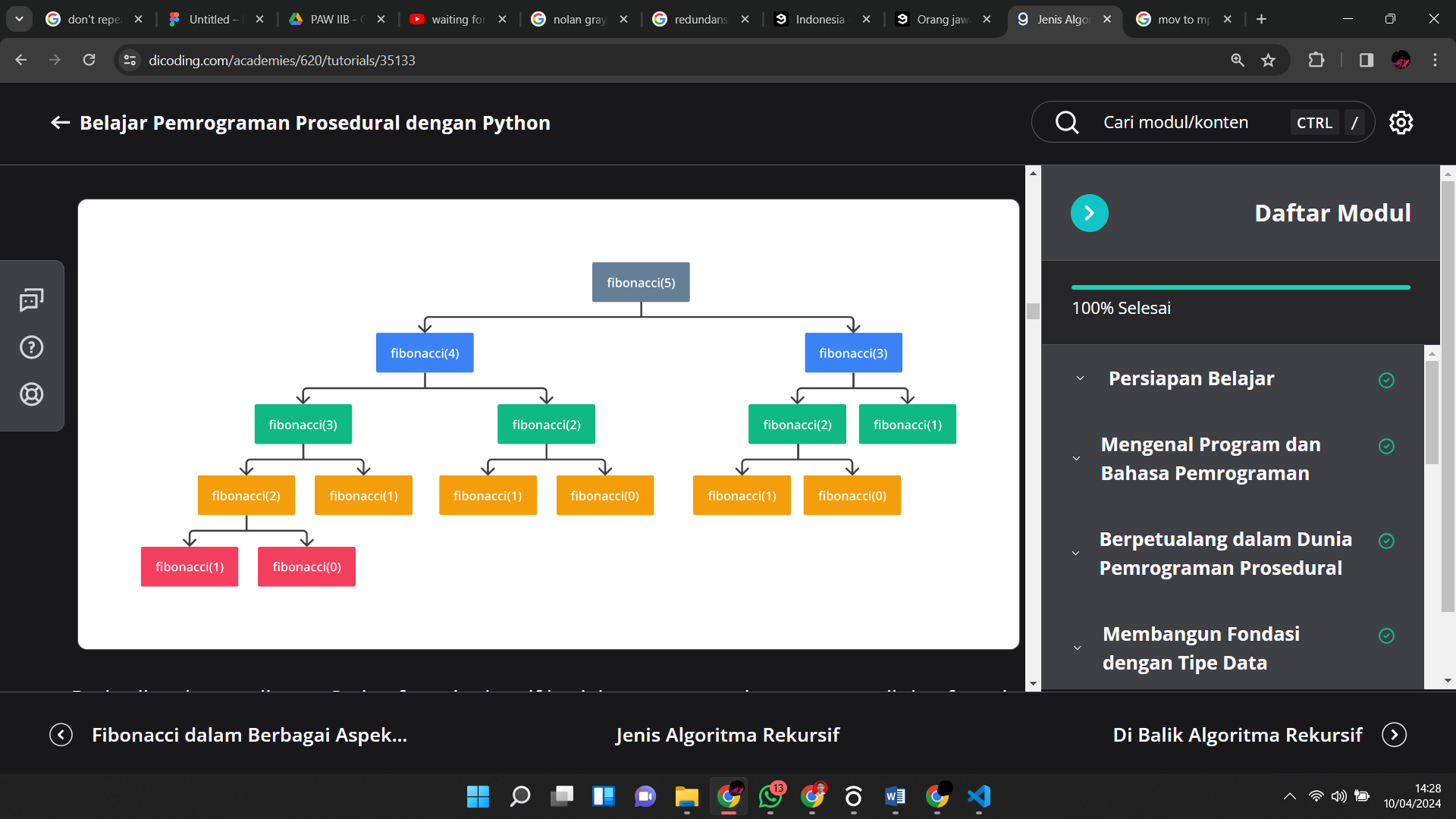
Maka output yang dihasilkan adalah





Dari hasil uraian tersebut, dapat dilihat bahwa setiap aksi hanya melakukan pemanggilan fungsi itu sendiri dan tidak ada proses rekursif yang lain. Proses yang dilalui cukup mudah dan hanya terbagi 2 aksi, yaitu mencetak angka dan memanggil fungsi itu kembali, hingga mencapai *base case* yang diinginkan.

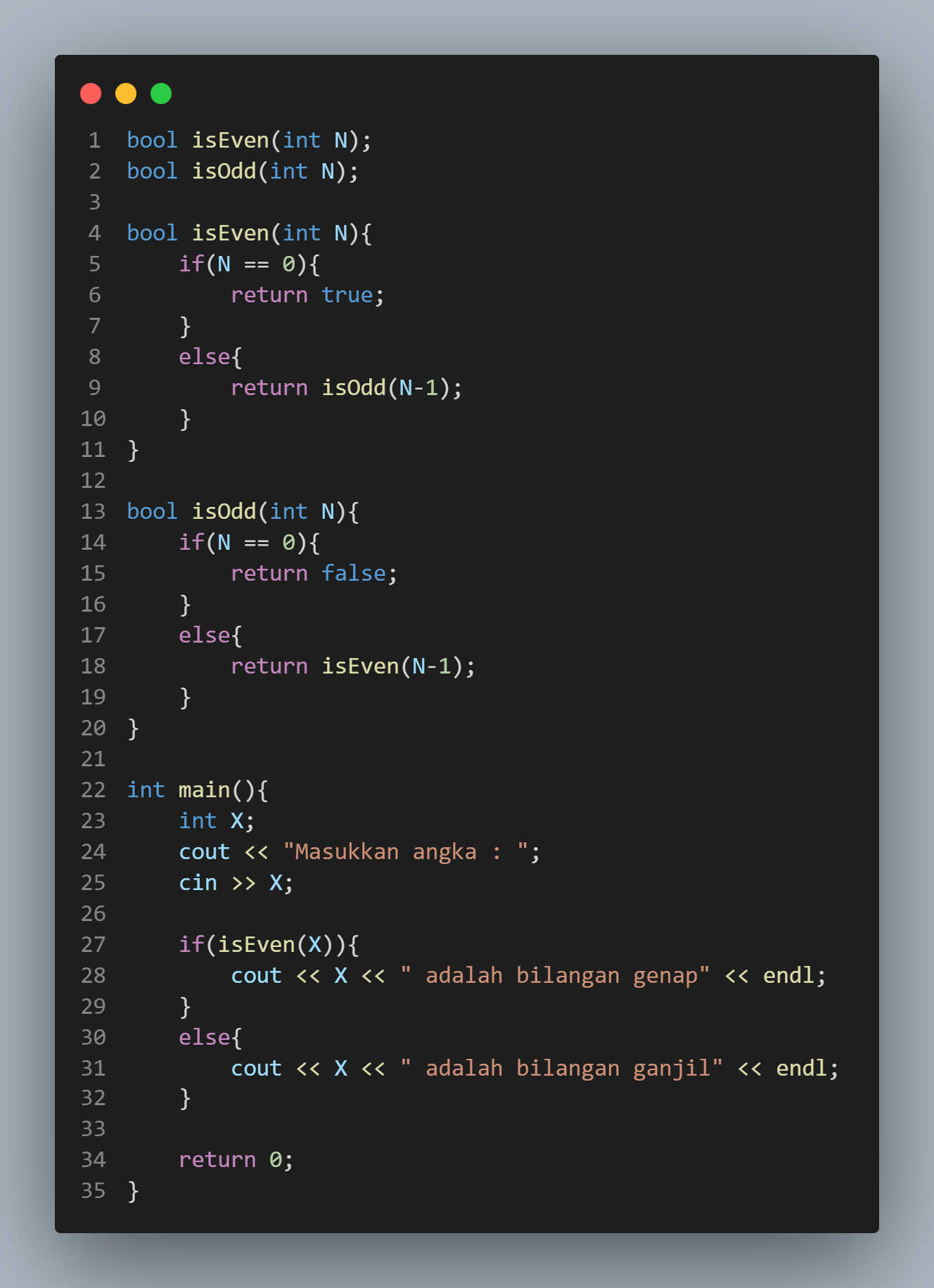
Sedangkan *tree recursion* adalah fungsi rekursif yang memanggil fungsi itu sendiri lebih dari satu kali. Contoh yang dapat dilihat adalah fungsi Fibonacci rekursif.



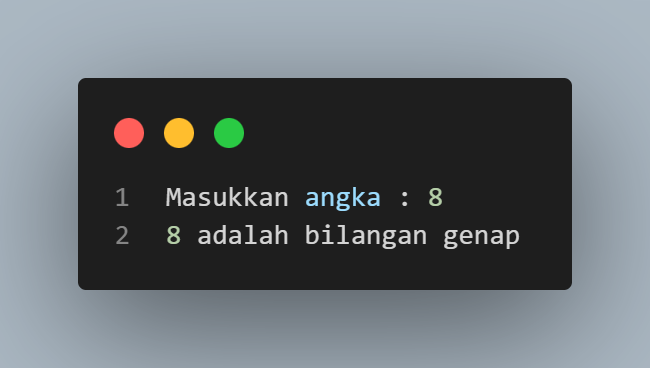
Dari hasil uraian diatas, terlihat menyerupai suatu pohon karena memiliki cabang yang banyak. Setiap fungsi rekursif dipanggil, maka program akan menjalankan 2 fungsi sekaligus, yaitu Fibonacci(n-1) dan Fibonacci(n-2). Tujuan dari 2 pemanggilan fungsi tersebut adalah untuk mengetahui 2 angka sebelum n, sekaligus untuk mengetahui apakah nilai sudah mendekati *base case* atau tidak.

Bila dilihat dari metode pemanggilan fungsi, fungsi terbagi menjadi 2 yaitu *direct recursion* dan *indirect recursion*. *Direct recursion* adalah pemanggilan fungsi rekursif secara langsung dan tidak ada perantara untuk proses pemanggilan fungsi. Singkatnya adalah fungsi yang memanggil dirinya sendiri.

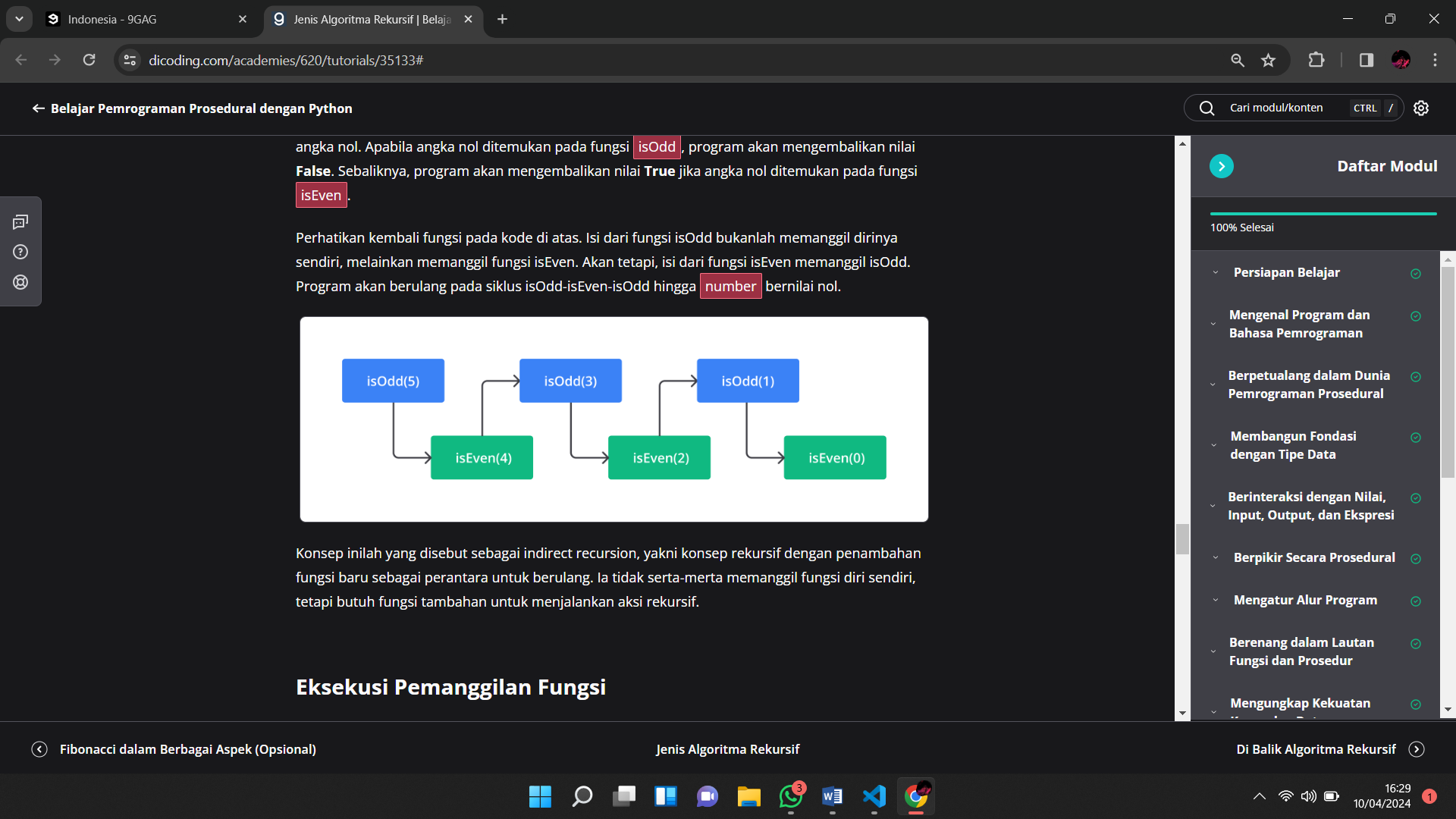
Sedangkan *indirect recursion* adalah pemanggilan fungsi rekursif secara tidak langsung dan ada perantara untuk proses pemanggilan fungsi. Biasanya memanggil fungsi yang lain, sehingga fungsi-fungsi tersebut saling melengkapi dan tidak dapat dipisahkan. Sebagai contoh fungsi IsOdd() dan IsEven()



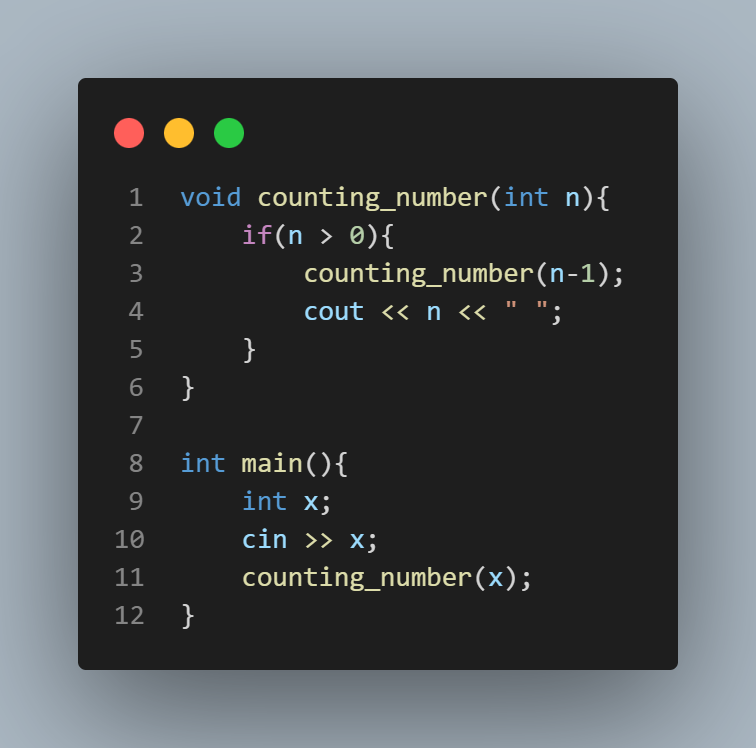
Maka output yang dihasilkan adalah



Konsep kerja dari program tersebut adalah untuk memeriksa apakah angka yang dimasukkan adalah angka ganjil atau genap. Jika angka belum mencapai 0, maka angka akan dikurangi dan dikembalikan ke fungsi yang lain. Apabila angka 0 ditemukan pada fungsi IsOdd(), maka angka tersebut adalah bilangan ganjil, tetapi apabila angka 0 ditemukan pada fungsi IsEven(), maka angka tersebut adalah bilangan genap. Sehingga, fungsi rekursif tidak selalu berbicara mengenai fungsi yang memanggil dirinya sendiri.



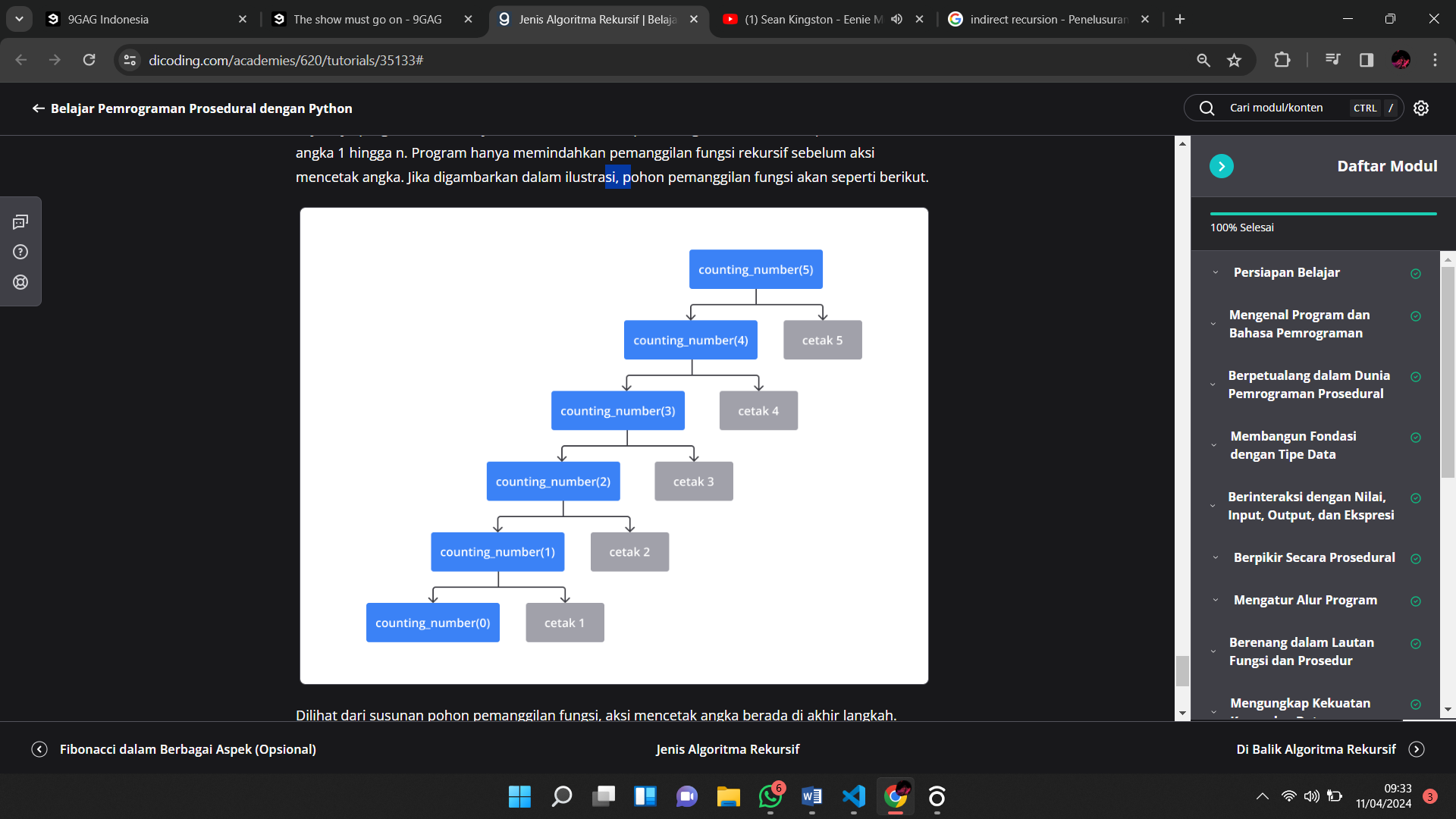
Fungsi rekursif juga bisa dibedakan dari letak eksekusi pemanggilan fungsi, yaitu *tail recursion* dan *non-tail recursion. Tail recursion* adalah fungsi rekursif yang pemanggilannya fungsinya berada di akhir blok fungsi. Kebanyakan fungsi rekursif menggunakkan konsep *tail recursion*. Sebagai contoh yaitu fungsi Fibonacci(), counting\_number(), isOdd(), dan isEven(). Sedangkan fungsi *non-tail recursion* adalah kebalikan dari *tail recursion*, dimana pemanggilannya berada di awal atau tengah blok fungsi. Sehingga tentunya logika dan algoritmanya akan berbeda dengan fungsi *tail recursion*. Sebagai contoh, berikut adalah fungsi counting\_number() yang menggunakkan konsep *non-tail recursion*



Maka output yang dihasilkan adalah



Hal ini bisa terjadi karena fungsi terus menerus melakukan pemanggilan fungsi sebelum mencetak angka. Sehingga angka yang diinputkan akan terus berkurang dan baru dicetak setelah angka mencapai batas, yaitu 0. Maka, output akan dimulai dari angka 1, bukan angka yang diinputkan. Berikut adalah penjelasan uraian untuk fungsi tersebut.



## 2.2 Pengimplementasian Fungsi Rekursif terhadap Kasus Permasalahan di Dunia Nyata

Andaikan terdapat suatu restoran menginginkan program yang bisa menyimpan data reservasi, menampilkan data reservasi, dan mencari nama pelanggan. Sehingga, setidaknya membutuhkan tiga fungsi untuk melaksanakan instruksi. Tapi, sebelum membuat tiga fungsi tersebut, terdapat 3 tipe data penting yang dibutuhkan, yaitu array untuk menyimpan data *customers*, *int Customers* untuk mencatat berapa banyak customers yang telah memesan. *Int MaxCustomers* untuk menjadi batas dari array.

int MaxCustomers = 10;

string Names[MaxCustomers] = {};

int Customers = 0;

Selanjutnya, kita bisa membuat *Dashboard Admin* untuk menambahkan reservasi, menampilkan daftar reservasi, mencari nama reservasi dan opsi keluar.

    char LoginAdmin = 'y';

        while(LoginAdmin == 'y'){

            int Input;

            system("cls");

            cout << "==== Dashboard Admin ====\n";

            cout << "1. Menambah Reservasi\n";

            cout << "2. Menampilkan Daftar Reservasi\n";

            cout << "3. Mencari Nama Reservasi\n";

            cout << "4. Keluar\n";

            cout << "> ";

            cin >> Input;

            cin.ignore();

Setiap pilihan telah memiliki logika, pemanggilan fungsi, argumentasi, tipe data yang dibutuhkan.

if(Input == 1){

      cout << endl;

      AddReservation(Names, Customers, MaxCustomers);

}

else if(Input == 2){

      cout << endl;

      cout << "Daftar Reservasi\n\n";

      ListReservation(Names, 0, Customers);

      cout << "\nKlik Enter untuk melanjutkan";

      cin.get();

      cout << endl;

}

else if(Input == 3){

      string TargetedName;

      cout << "Masukkan nama yang ingin dicari: ";

      getline(cin, TargetedName);

      bool Found = FindReservation(Names, 0, 9, TargetedName);

      if (Found){

            cout << endl << TargetedName << " telah memesan meja\n\n";

      }

else{

            cout << endl << TargetedName << " belum memesan meja\n\n";

      }

cout << "Klik Enter untuk melanjutkan";

      cin.get();

}

else if(Input == 4){

      char Exit;

      cout << "Apakah anda ingin keluar dari akses Admin (y/n)\n";

      cout << "> ";

      cin >> Exit;

      if(Exit == 'y' || Exit == 'Y'){

          LoginAdmin = 'n';

       }

       else{

          LoginAdmin = 'y';

       }

    }

}

return 0;

}

Fungsi 1 : Menambah reservasi

void AddReservation(string Names[],int& Customers, const int& MaxCustomers){

    char AddCustomer;

    string NewName;

    cout << "Masukkan nama pelanggan baru\n";

    cout << "> ";

    getline(cin, NewName);

    if(Customers < MaxCustomers){

        Names[Customers] = NewName;

        Customers++;

    }

    else{

        cout << "Meja telah penuh\n";

return;

    }

    cout << "Ingin menambah reservasi lagi? (y/n)\n";

    cout << "> ";

    cin >> AddCustomer;

    cin.ignore();

    if(AddCustomer == 'y'){

        AddReservation(Names, Customers, MaxCustomers);

    }

}

Fungsi *AddReservation()* menerima 3 parameter, array *string* *Names*, *int reference* *Customers*, dan *constant int reference MaxCustomers*. Ketiga parameter tersebut digunakan agar fungsi bisa mengakses data dari 3 parameter tersebut*. String NewName* dibuat untuk memasukkan nama pelanggan baru. Lalu dilakukan pengecekan, apakah jumlah *Customers* tidak lebih banyak dari *MaxCustomers*, yang merupakan batas dari *customers*.

Jika *Customers* kurang dari batas *Customer,* yaitu 10, maka nama pelanggan baru dimasukkan dalam array. *Int Customers* ditambahkan juga agar mengetahui berapa banyak *customer* yang telah melakukan reservasi dan bisa dimasukkan ke kolom array berikutnya.

Untuk proses rekursif dari fungsi tersebut dapat dilihat dari bagian fungsi paling bawah. Apabila *user* ingin menambahkan data kembali dengan menginputkan ‘y’, maka fungsi akan memanggil kembali fungsi *AddReservation*. Untuk *base case* dari fungsi tersebut apabila user menginputkan ‘n’ pada bagian “Ingin menambah reservasi lagi?(y/n) >”.

Fungsi 2 : Menampilkan data reservasi

void ListReservation(string Names[],int Stack, int& Customers){

    if(Stack == Customers){

        return;

    }

    cout << "Meja " << Stack+1 << ", atas nama " << Names[Stack] << endl;

    ListReservation(Names,Stack+1,Customers);

}

Fungsi *ListReservation()* menerima 3 parameter, array *string* *Names*, *int Stack*, dan *int reference Customers*. Ketiga parameter tersebut digunakan agar fungsi bisa mengakses data dari 3 parameter tersebut. Fungsi akan mencetak nama-nama yang ada di dalam array dan setiap anggota array akan diakses oleh *int Stack*.

Untuk proses rekursif dari fungsi tersebut dapat dilhat pada bagian fungsi paling bawah. Apabila *Stack* nilainya masih kurang dari *int Customers* (yang merupakan *counter* dari banyaknya reservasi), maka rekursif akan terus dilakukan. Sehingga, *base case* dari fungsi tersebut adalah apabila nilai dari *Stack* sama dengan nilai dari *Customers*.

Fungsi 3 : Mencari nama reservasi

bool FindReservation(string Names[], int Start, int End, const string& TargetedName){

    if(Start > End){

        return false;

    }

    if(Names[Start] == TargetedName){

        return true;

    }

    return FindReservation(Names, Start + 1, End - 1, TargetedName);

}

Fungsi *FindReservation()* menerima 4 parameter, array *string Names, int Start, int End, constant string reference TargetedName*. Fungsi tersebut akan mengembalikan nilai *true* atau *false.* Tujuan dari dibuatnya *int Start* dan *int End* adalah menjadi *counter* dalam pengaksesan array *Names*.

Apabila di dalam *array Names* terdapat nama yang diinputkan *user*. Maka fungsi akan mengembalikan nilai *true,* sehingga nama tersebut telah melakukan reservasi. Jika memang tidak, maka proses rekursif akan dilakukan dengan argumen yang sama. Hanya saja *int Start* ditambahkan 1 dan *int End* dikurangi 1. *Base case* dari fungsi tersebut apabila nilai *Start* melebihi nilai *End*. Berarti nama tersebut belum melakukan reservasi dan fungsi mengembalikan nilai *false*.

Sehingga, apabila semua program tersebut digabung, maka terciptalah program seperti berikut

#include <iostream>

using namespace std;

int MaxCustomers = 10;

void AddReservation(string Names[],int& Customers, const int& MaxCustomers){

    char AddCustomer;

    string NewName;

    cout << "Masukkan nama pelanggan baru\n";

    cout << "> ";

    getline(cin, NewName);

    if(Customers < MaxCustomers){

        Names[Customers] = NewName;

        Customers++;

    }

    else{

        cout << "Meja telah penuh\n";

return;

}

    cout << "Ingin menambah reservasi lagi? (y/n)\n";

    cout << "> ";

    cin >> AddCustomer;

    cin.ignore();

    if(AddCustomer == 'y'){

        AddReservation(Names, Customers, MaxCustomers);

    }

}

void ListReservation(string Names[],int Stack, int& Customers){

    if(Stack == Customers){

        return;

    }

    cout << "Meja " << Stack+1 << ", atas nama " << Names[Stack] << endl;

    ListReservation(Names,Stack+1,Customers);

}

bool FindReservation(string Names[], int Start, int End, const string& TargetedName) {

    if(Start > End){

        return false;

    }

    if(Names[Start] == TargetedName){

        return true;

    }

    return FindReservation(Names, Start + 1, End - 1, TargetedName);

}

int main() {

    string Names[MaxCustomers] = {};

    int Customers = 0;

    char LoginAdmin = 'y';

        while(LoginAdmin == 'y'){

            int Input;

            system("cls");

            cout << "==== Dashboard Admin ====\n";

            cout << "1. Menambah Reservasi\n";

            cout << "2. Menampilkan Daftar Reservasi\n";

            cout << "3. Mencari Nama Reservasi\n";

            cout << "4. Keluar\n";

            cout << "> ";

            cin >> Input;

            cin.ignore();

            if(Input == 1){

              cout << endl;

              AddReservation(Names, Customers, MaxCustomers);

            }

          else if(Input == 2){

              cout << endl;

              cout << "Daftar Reservasi\n\n";

              ListReservation(Names, 0, Customers);

              cout << "\nKlik Enter untuk melanjutkan";

              cin.get();

            cout << endl;

           }

        else if(Input == 3){

              string TargetedName;

              cout << "Masukkan nama yang ingin dicari: ";

              getline(cin, TargetedName);

              bool Found = FindReservation(Names, 0, 9, TargetedName);

            if (Found) {

            cout << endl << TargetedName << " telah memesan meja\n\n";

              } else {

                  cout << endl << TargetedName << " belum memesan meja\n\n";

              }

      cout << "Klik Enter untuk melanjutkan";

             cin.get();

          }

          else if(Input == 4){

              char Exit;

              cout << "Apakah anda ingin keluar dari akses Admin (y/n)\n";

              cout << "> ";

              cin >> Exit;

              if(Exit == 'y' || Exit == 'Y'){

                 LoginAdmin = 'n';

              }

              else{

                LoginAdmin = 'y';

              }

          }

      }

    return 0;

}

Sehingga, terlihat bahwa semua fungsi menggunakkan fungsi rekursif.

## 2.3 Perbandingan antara Fungsi Biasa dengan Fungsi Rekursif

Apabila semua fungsi rekursif dirubah menjadi fungsi biasa yang menggunakkan iterasi. Maka berikut adalah programnya

void AddReservation(string Names[],int& Customers, const int& MaxCustomers){

    char AddCustomer = 'y';

    while(AddCustomer == 'y'){

        string NewName;

        cout << "Masukkan nama pelanggan baru\n";

        cout << "> ";

        getline(cin, NewName);

        if(Customers < MaxCustomers){

            Names[Customers] = NewName;

            Customers++;

        }

        else{

            cout << "Meja telah penuh\n";

            return;

        }

        cout << "Ingin menambah reservasi lagi? (y/n)\n";

        cout << "> ";

        cin >> AddCustomer;

        cin.ignore();

    }

}

void ListReservation(string Names[],int& Customers){

    for(int i = 0; i < Customers; i++){

    cout << "Meja " << i+1 << ", atas nama " << Names[i] << endl;

    }

}

bool FindReservation(string Names[], int& Customers, const string& TargetedName) {

    for(int i = 0; i < Customers; i++){

    if(Names[i] == TargetedName){

        return true;

        }

    }

    return false;

}

int main() {

    string Names[MaxCustomers] = {};

    int Customers = 0;

    char LoginAdmin = 'y';

        while(LoginAdmin == 'y'){

            int Input;

            system("cls");

            cout << "==== Dashboard Admin ====\n";

            cout << "1. Menambah Reservasi\n";

            cout << "2. Menampilkan Daftar Reservasi\n";

            cout << "3. Mencari Nama Reservasi\n";

            cout << "4. Keluar\n";

            cout << "> ";

            cin >> Input;

            cin.ignore();

            if(Input == 1){

              cout << endl;

              AddReservation(Names, Customers, MaxCustomers);

            }

          else if(Input == 2){

              cout << endl;

              cout << "Daftar Reservasi\n\n";

              ListReservation(Names, 0, Customers);

              cout << "\nKlik Enter untuk melanjutkan";

              cin.get();

            cout << endl;

           }

        else if(Input == 3){

              string TargetedName;

              cout << "Masukkan nama yang ingin dicari: ";

              getline(cin, TargetedName);

              bool Found = FindReservation(Names, 0, 9, TargetedName);

            if (Found) {

            cout << endl << TargetedName << " telah memesan meja\n\n";

              } else {

                  cout << endl << TargetedName << " belum memesan meja\n\n";

              }

      cout << "Klik Enter untuk melanjutkan";

             cin.get();

          }

          else if(Input == 4){

              char Exit;

              cout << "Apakah anda ingin keluar dari akses Admin (y/n)\n";

              cout << "> ";

              cin >> Exit;

              if(Exit == 'y' || Exit == 'Y'){

                 LoginAdmin = 'n';

              }

              else{

                LoginAdmin = 'y';

              }

          }

      }

    return 0;

}

## 2.4 Output yang dihasilkan dari Fungsi Rekursif

==== Dashboard Admin ====

1. Menambah Reservasi

2. Menampilkan Daftar Reservasi

3. Mencari Nama Reservasi

4. Keluar

> 1

Masukkan nama pelanggan baru

> Louis Nathan

Ingin menambah reservasi lagi? (y/n)

> y

Masukkan nama pelanggan baru

> Luki Ahmad

Ingin menambah reservasi lagi? (y/n)

> y

Masukkan nama pelanggan baru

> Muhammad Febryan

Ingin menambah reservasi lagi? (y/n)

> n

==== Dashboard Admin ====

1. Menambah Reservasi

2. Menampilkan Daftar Reservasi

3. Mencari Nama Reservasi

4. Keluar

> 2

Daftar Reservasi

Meja 1, atas nama Louis Nathan

Meja 2, atas nama Luki Ahmad

Meja 3, atas nama Muhammad Febryan

Klik Enter untuk melanjutkan

==== Dashboard Admin ====

1. Menambah Reservasi

2. Menampilkan Daftar Reservasi

3. Mencari Nama Reservasi

4. Keluar

> 3

Masukkan nama yang ingin dicari: Luki Ahmad

Luki Ahmad telah memesan meja

Klik Enter untuk melanjutkan

==== Dashboard Admin ====

1. Menambah Reservasi

2. Menampilkan Daftar Reservasi

3. Mencari Nama Reservasi

4. Keluar

> 3

Masukkan nama yang ingin dicari: Muhammad Jack

Muhammad Jack belum memesan meja

Klik Enter untuk melanjutkan

==== Dashboard Admin ====

1. Menambah Reservasi

2. Menampilkan Daftar Reservasi

3. Mencari Nama Reservasi

4. Keluar

> 4

Apakah anda ingin keluar dari akses Admin (y/n)

> y