

LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN
“POSTEST PRAKTIKUM 5: REKURSI”

Diajukan untuk memenuhi salah satu praktikum Mata Kuliah Algoritma Pemrograman yang
di ampu oleh:

Dr. Ardiansyah S.T., M.Cs



Disusun Oleh:

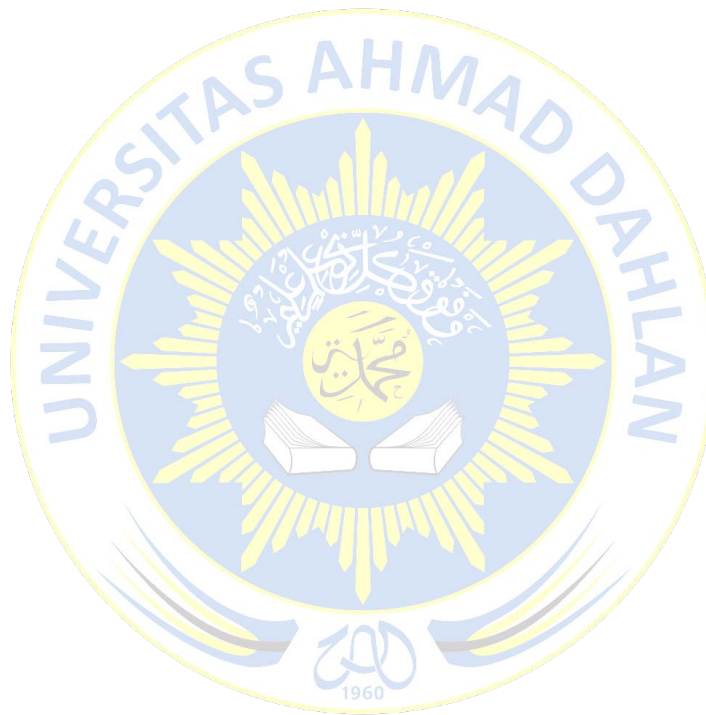
Mohammad Farid Hendianto 2200018401

A / Jumat 13.30 Lab. Jaringan

PROGRAM STUDI INFORMATIKA
UNIVERSITAS AHMAD DAHLAN
FAKULTAS TEKNOLOGI INDUSTRI
TAHUN 2023

DAFTAR SOAL

1. Buat lah flowchart untuk membuat fungsi rekursif untuk menyelesaikan deret dibawah ini: 3
2. Seperti nomor 1, gunakan subprogram dalam flowchart untuk membuat fungsi rekursif untuk menyelesaikan deret dibawah ini: 6
3. Konversikan hasil dari flowchart nomor 1 dan 2 menjadi program C++. 9



1. Buat lah flowchart untuk membuat fungsi rekursif untuk menyelesaikan deret dibawah ini:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{n}$$

Jawab:

$$\text{Rumus deret tersebut adalah } F(n) = (-1)^{n+1} \times \frac{1}{n}$$

Sebelum menjelaskan tentang fungsi rekursif dan flowchart, saya akan terlebih dahulu menjelaskan konsep dari deret tersebut. Deret yang disebut adalah deret alternatif, yang artinya setiap suku dalam deret tersebut bergantian antara positif dan negatif. Dalam kasus ini, kita diminta untuk menemukan jumlah deret tersebut sampai suku ke-n. Fungsi $F(n)$ diberikan untuk membantu menghitung jumlah tersebut.

Fungsi $F(n) = (-1)^{(n+1)} * 1/n$ memungkinkan kita untuk menentukan apakah suku yang sedang dihitung pada saat itu harus dikurangi atau ditambahkan ke hasil sebelumnya. Nilai $(-1)^{(n+1)}$ akan memberikan nilai positif jika n ganjil dan nilai negatif jika n genap. Kemudian, $1/n$ akan digunakan untuk menghitung jumlah selanjutnya dalam deret tersebut.

Sekarang, mari kita bahas tentang fungsi rekursif. Fungsi rekursif adalah sebuah fungsi yang memanggil dirinya sendiri secara berulang-ulang hingga mencapai kondisi dasar atau base case. Base case merupakan kondisi di mana fungsi tidak lagi memanggil dirinya sendiri dan mengembalikan nilai yang diinginkan. Konsep ini sering digunakan pada struktur data seperti linked list dan binary tree serta algoritma seperti quicksort dan merge sort.

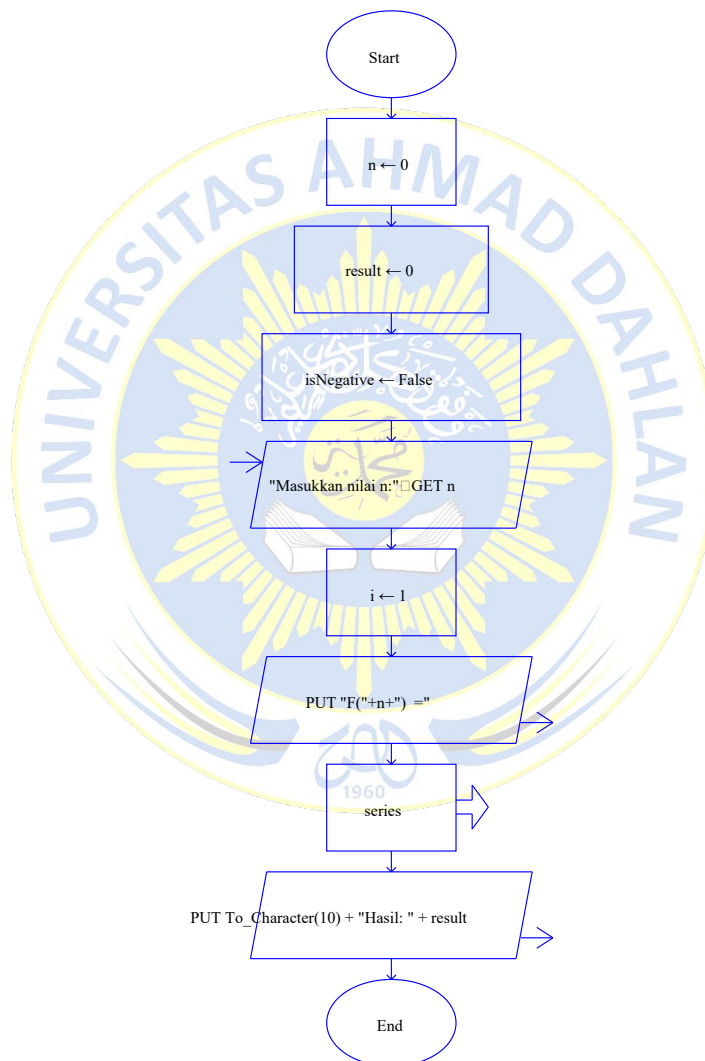
Pada flowchart yang telah diberikan, pengguna diminta untuk memasukkan nilai n . Selanjutnya, variabel n akan menjadi kondisi untuk melakukan looping melalui deret tersebut menggunakan variabel i sebagai penghitung. Pada setiap iterasi, fungsi akan mengecek apakah suku tersebut harus ditambahkan atau dikurangi dari hasil sebelumnya menggunakan variabel $isNegative$. Kemudian, nilai suku tersebut akan dihitung berdasarkan nilai i dan dimasukkan ke dalam variabel $result$. Setelah itu, variabel i akan ditambah 1 untuk melanjutkan ke suku selanjutnya.

Jika nilai i masih kurang dari atau sama dengan n , maka fungsi akan memanggil dirinya sendiri dengan nilai i yang telah ditingkatkan. Jika sudah mencapai base case, maka hasil akhir akan dicetak pada layar.

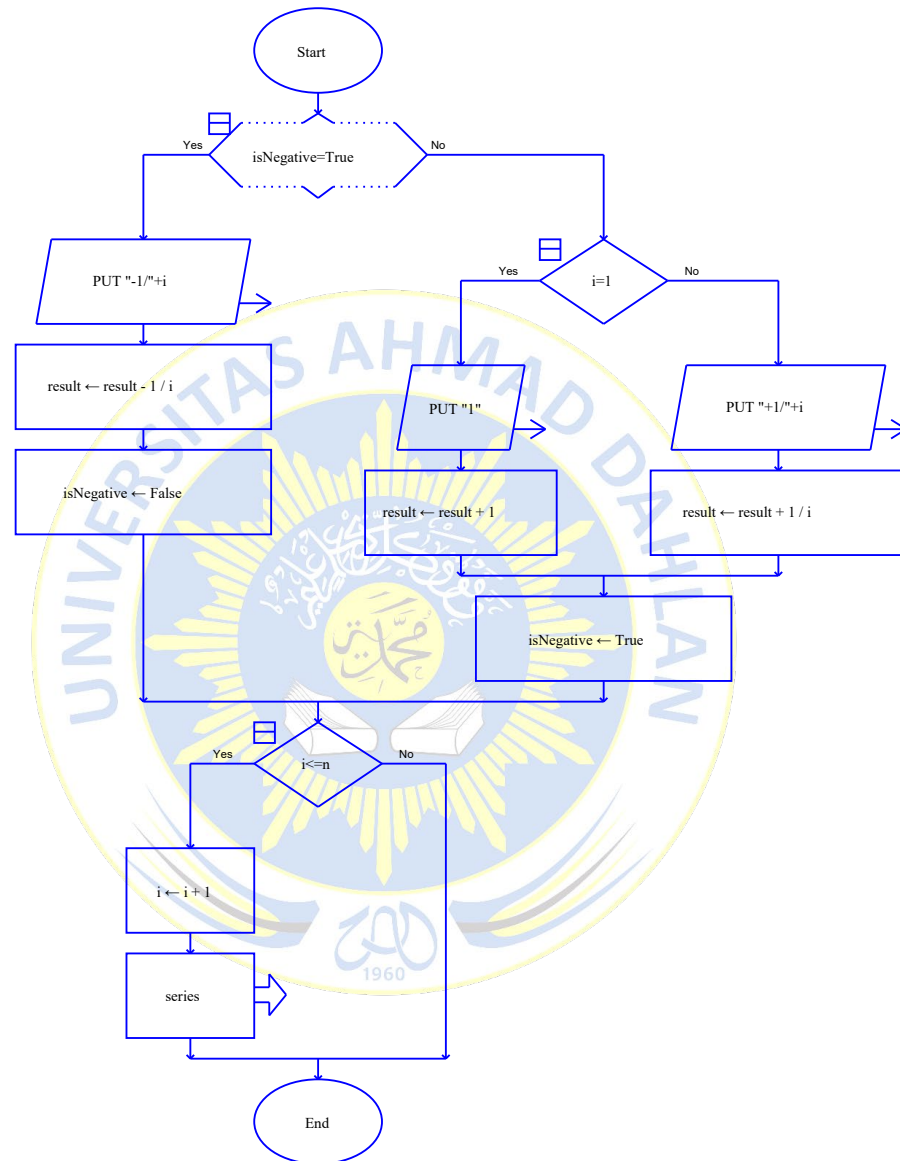
Namun, perlu diperhatikan bahwa flowchart yang diberikan **belum sepenuhnya menggunakan konsep rekursi**. Dalam flowchart tersebut, ketika fungsi dipanggil kembali, nilai variabel i akan direset menjadi 1 dan variabel $result$ akan direset menjadi 0. Hal ini berarti fungsi tidak akan menyimpan nilai sebelumnya dan hanya menghitung hasil terakhir saja. Untuk mengubahnya menjadi fungsi rekursif yang benar, kita perlu menambahkan parameter yang menyimpan nilai sebelumnya dan mengembalikan nilai hasil saat mencapai base case.

Dalam penggunaan subFlowchart seperti yang dijelaskan dalam flowchart, fungsi rekursif tidak secara eksplisit memanggil dirinya sendiri, namun menggunakan subFlowchart sebagai pengganti pemanggilan fungsi rekursif. Ini bisa dilakukan dengan membuat subFlowchart baru yang menerima parameter yang sama dengan fungsi rekursif dan melakukan hal yang sama namun dengan cara yang sedikit berbeda.

Berikut adalah gambar flowchart untuk membuat fungsi rekursif untuk menyelesaikan deret di atas:



Gambar 1 Fungsi utama. (Sumber: Penulis)



Gambar 2 Menggunakan subFlowchart series sebagai alternatif fungsi agar bisa menggunakan fungsi rekursif.
(Sumber: Penulis)

Secara umum, alur dalam flowchart ini dapat dijelaskan sebagai berikut:

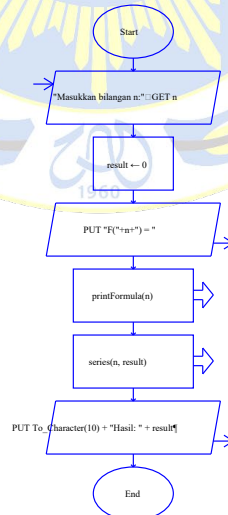
- 1) Inisialisasi variabel n, result, i, isnegative dengan nilai awal 0 atau false.
- 2) Meminta pengguna untuk memasukkan nilai n.
- 3) Menampilkan output "F(n) = ".
- 4) Jika nilai isnegative true, maka tampilkan "-1/i" dan kurangi result dengan nilai 1/i. Kemudian ubah nilai isnegative menjadi false.
- 5) Jika nilai i sama dengan 1, maka tampilkan "1" dan tambahkan result dengan nilai 1.
- 6) Jika nilai i lebih besar dari 1, maka tampilkan "+1/i" dan tambahkan result dengan nilai 1/i. Ubah nilai isnegative menjadi true.
- 7) Tambahkan nilai i dengan 1.
- 8) Jika nilai i masih kurang dari atau sama dengan n, panggil kembali fungsi seriesFormula.
- 9) Jika nilai i lebih besar daripada n, tampilkan hasilnya dengan menambahkan karakter baris baru dan output "Hasil: " diikuti dengan nilai result.

Berikut adalah outputnya

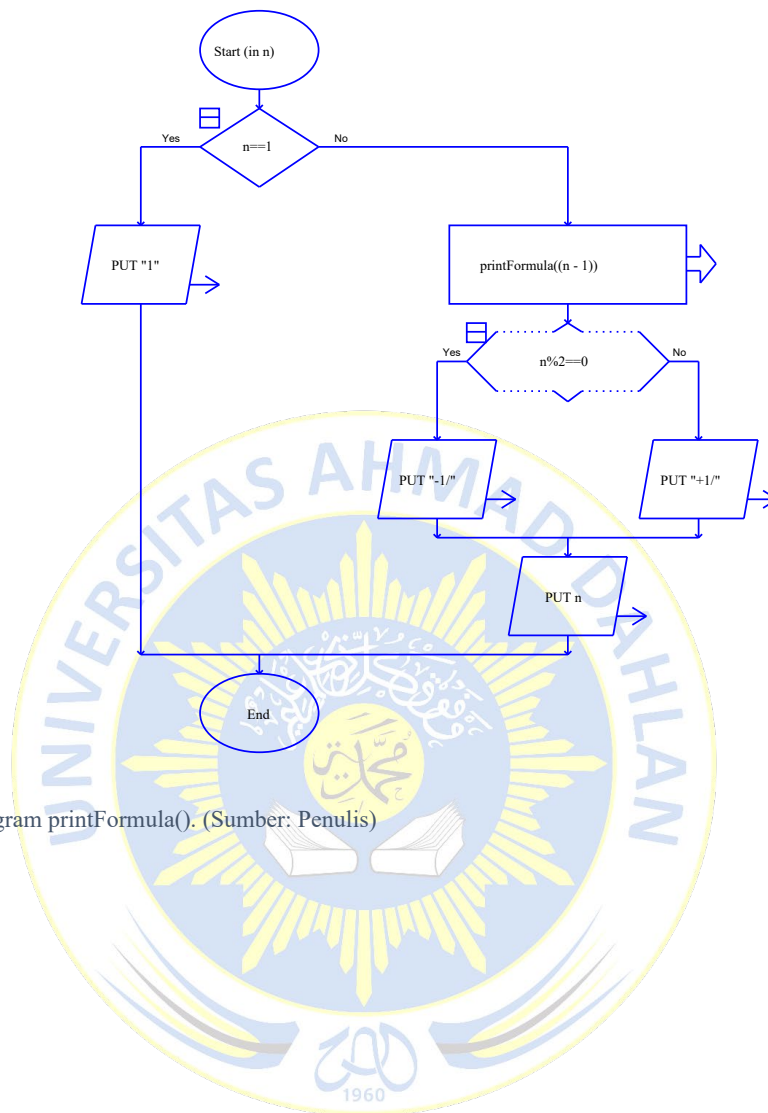
2. Seperti nomor 1, gunakan subprogram dalam flowchart untuk membuat fungsi rekursif untuk menyelesaikan deret dibawah ini:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{n}$$

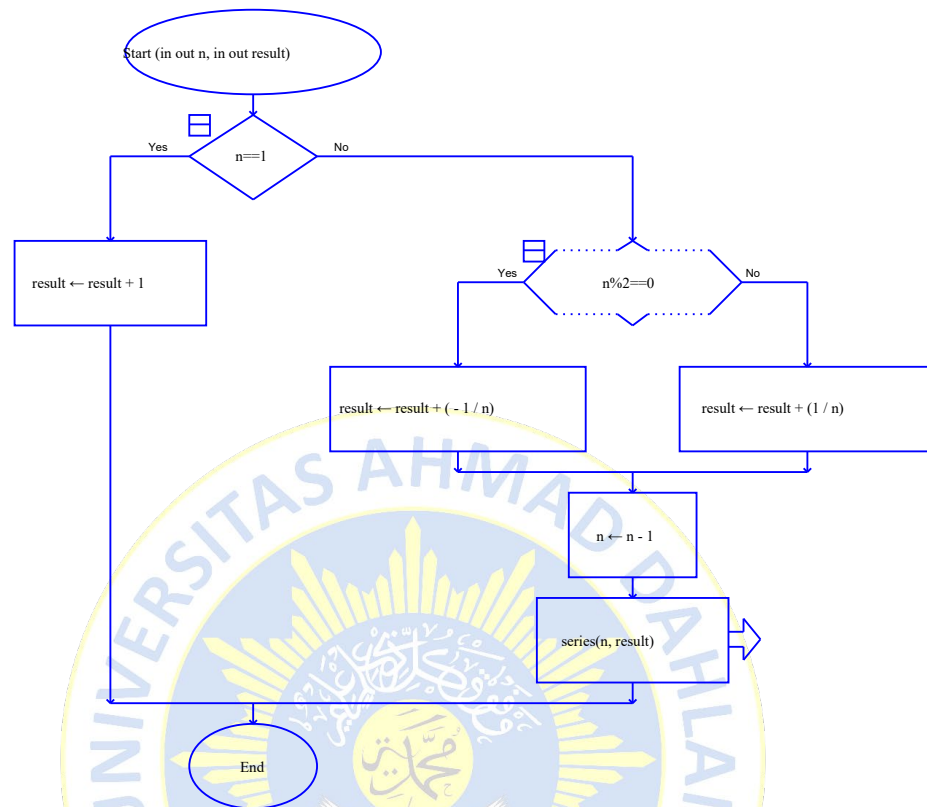
Jawab:



Gambar 3 Flowchart utama. (Sumber: Penulis)



Gambar 4 Subprogram printFormula(). (Sumber: Penulis)



Gambar 5 Subprogram series(). (Sumber: Penulis)

Flowchart di atas merupakan suatu subprogram dalam bahasa pemrograman yang bertujuan untuk menghitung deret bilangan dengan formula tertentu sebanyak n kali.

Dalam subprogram ini, terdapat dua subprosedur yaitu printFormula dan series. Prosedur printFormula berfungsi untuk menampilkan formula dari deret bilangan yang akan dihitung, sedangkan prosedur series berfungsi untuk menghitung hasil dari deret bilangan tersebut. Kedua prosedur ini dipanggil secara berulang dalam subprogram hingga seluruh deret bilangan berhasil dihitung.

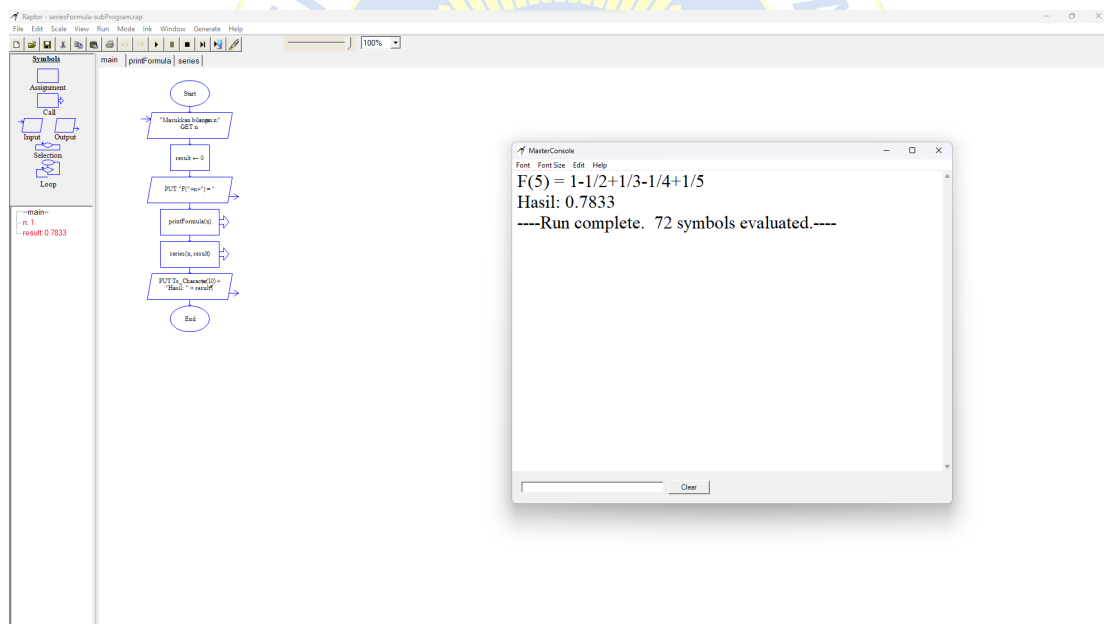
Pada awal subprogram dijalankan, user akan diminta untuk memasukkan nilai n sebagai panjang jumlah deret bilangan yang akan dihitung. Kemudian, program akan menjalankan prosedur printFormula untuk menampilkan formula dari deret bilangan yang akan dihitung. Proses printFormula akan mengecek apakah nilai n bernilai 1 atau tidak. Jika iya, maka akan menampilkan angka 1. Jika tidak, maka prosedur akan melakukan rekursif dengan memanggil dirinya sendiri dengan parameter $(n-1)$. Selama proses rekursif dilakukan, program akan mengecek apakah nilai n adalah bilangan genap atau ganjil. Jika genap, maka program akan menampilkan tanda negatif dan pecahan satu dibagi n . Jika ganjil, maka program akan

menampilkan tanda positif dan pecahan satu dibagi n. Proses ini akan terus berulang hingga nilai n sama dengan 1.

Setelah nilai formula ditampilkan, program akan menjalankan prosedur series untuk menghitung hasil dari deret bilangan tersebut. Prosedur series akan mengecek apakah nilai n bernilai 1 atau tidak. Jika iya, maka program akan menambahkan angka 1 pada hasil deret bilangan. Jika tidak, maka program akan mengecek apakah nilai n adalah bilangan genap atau ganjil. Jika genap, maka program akan menambahkan $(-1/n)$ pada hasil deret bilangan. Jika ganjil, maka program akan menambahkan $(1/n)$ pada hasil deret bilangan. Setelah itu, nilai n akan dikurangi 1 dan prosedur series akan rekursif dengan memanggil dirinya sendiri dengan parameter $(n-1)$ dan hasil deret bilangan yang telah dihitung sebelumnya.

Setelah semua proses selesai, program akan menampilkan hasil dari deret bilangan yang telah dihitung. Proses ini dilakukan dengan cara menampilkan karakter newline dan kata "Hasil: " yang diikuti dengan nilai dari variabel result.

Berikut adalah outputnya:



Gambar 6 Output jika series(5). (Sumber: Penulis)

3. Konversikan hasil dari flowchart nomor 1 dan 2 menjadi program C++.

Konversi Nomor 1:

Berikut adalah hasil konversi flowchart nomor 1, tetapi bukan sepenuhnya menggunakan fungsi rekursif karena fungsi rekursif harus menggunakan subprogram (tetapi disini menggunakan subFlowchart).

```

1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int n;
6      double result = 0.0;
7      bool isNegative = false;
8
9      cout << "Masukkan nilai n: ";
10     cin >> n;
11     cout << "F(" << n << ") = ";
12     int i=1;
13
14     series: // subflowchart series
15     while (i<=n){
16         if(isNegative){
17             cout<<" - 1/"<<i;
18             result -= 1.0/i;
19             isNegative = false;
20         }else{
21             if(i==1){
22                 cout<<1;
23                 result += 1;
24             }else{
25                 cout<<" + 1/"<<i;
26                 result += 1.0/i;
27             }
28             isNegative = true;
29         }
30         i++;
31         goto series;
32     }
33
34     cout << "\nHasil: " << result << endl;
35 }

```

Gambar 7 Hasil konversi flowchart nomor 1. (Sumber: Penulis)

Berikut adalah outputnya:

```

1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int n;
6     double result = 0.0;
7     bool isNegative = false;
8
9     cout << "Masukkan nilai n: ";
10    cin >> n;
11    cout << "F(" << n << ") = ";
12    int i=1;
13
14    series: // subflowchart series
15    while (i<=n){
16        if(isNegative){
17            cout<< " - 1/"<<i;
18            result -= 1.0/i;
19            isNegative = false;
20        }
21        else{
22            cout<< " + 1/"<<i;
23            result += 1.0/i;
24            isNegative = true;
25        }
26        i++;
27    }
28    cout << result;
29    return 0;
30 }

```

```

PS C:\Users\Ndik> cd "d:\Document Ndik\Kuliah\Semester 2\Algoritma Pemrograman\Praktikum\Pertemuan 5"; if ($?) { g++ seriesFormula_subFlowchart.cpp -o seriesFormula_subFlowchart }; if ($?) { .\seriesFormula_subFlowchart }
Masukkan nilai n: 5
F(5) = 1 - 1/2 + 1/3 - 1/4 + 1/5
Hasil: 0.783333
PS D:\Document Ndik\Kuliah\Semester 2\Algoritma Pemrograman\Praktikum\Pertemuan 5>

```

Gambar 8 Output jika n=5. (Sumber: Penulis)

Kodingan ini adalah program untuk menghitung deret bilangan $F(n)$ dengan menggunakan formula:

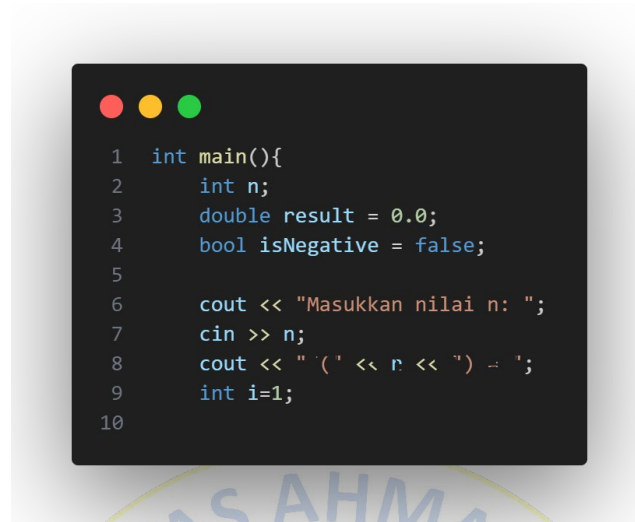
$$F(n) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{(-1)^{n+1}}{n}$$

Program akan meminta input nilai n dari pengguna, kemudian menghitung $F(n)$ dengan menggunakan rumus tersebut. Program akan menampilkan deret bilangan yang digunakan untuk menghitung $F(n)$, serta nilai $F(n)$ itu sendiri.



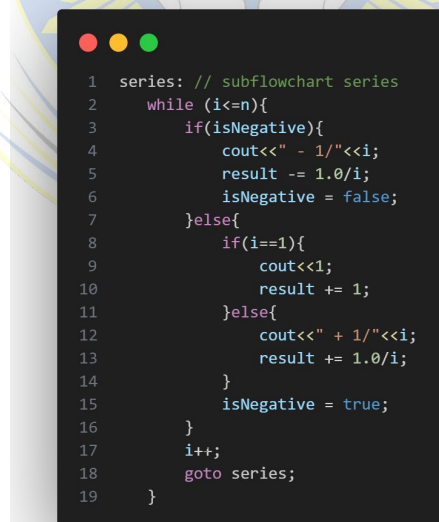
Gambar 9 Header. (Sumber: Penulis)

Baris pertama adalah directive `#include` yang digunakan untuk menyertakan header file `iostream`. Header file ini berisi fungsi-fungsi input/output pada C++, seperti `cout`, `cin`, dan lain-lain. Sedangkan baris kedua menggunakan namespace `std`.



Gambar 10 Inisialisasi variabel dan input n. (Sumber: Penulis)

Fungsi main() adalah fungsi utama dalam program C++. Pada bagian ini, variabel-variabel yang dibutuhkan disiapkan. Variabel n bertipe integer digunakan untuk menyimpan jumlah elemen dalam deret. Variabel result bertipe double digunakan untuk menyimpan hasil dari perhitungan $F(n)$. Variabel boolean isNegative akan digunakan untuk menentukan apakah elemen deret saat ini negatif atau tidak. Kemudian, program akan meminta input dari pengguna untuk nilai n dan menampilkan " $F(n) =$ ", serta menginisialisasi variabel i dengan 1.



Gambar 11 Hasil Subflowchart series(). (Sumber: Penulis)

Bagian ini adalah subflowchart series. Program akan melakukan perulangan sebanyak n kali menggunakan while, dimulai dari $i=1$ hingga i sama dengan n . Setiap iterasi, program akan mengecek apakah elemen saat ini negatif atau tidak. Jika `isNegative` bernilai `true`, maka program akan menampilkan `"- 1/i"` dan mengurangi nilai `result` dengan 1 dibagi i . Jika `isNegative` bernilai `false`, maka program akan menampilkan `" + 1/i"` dan menambahkan nilai `result` dengan 1 dibagi i . Setelah itu, nilai `isNegative` akan diubah menjadi kebalikan nilainya dengan operator `!`. Variabel `i` kemudian diincrement dan program akan melompat kembali ke label `series`.

Pada awalnya, `goto` seringkali dianggap sebagai praktik yang buruk dalam pemrograman karena bisa menyebabkan spaghetti code atau tak terkendali. Namun, pada baris kode di atas, penggunaan `goto` dipilih karena program harus melakukan perulangan dengan logika yang agak rumit dan tidak bisa diimplementasikan dengan mudah menggunakan struktur kontrol seperti `for` atau `while`.

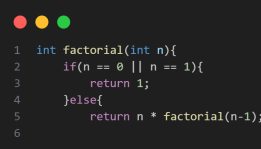


```
1 cout << "\nHasil: " << result << endl;
2
```

Gambar 12 Hasil output di `result`. (Sumber: Penulis)

Terakhir, program akan menampilkan hasil perhitungan $F(n)$ yang disimpan dalam variabel `result`.

Sekarang, mari kita bahas mengapa program ini tidak sepenuhnya menggunakan fungsi rekursif. Pada umumnya, fungsi rekursif adalah fungsi yang memanggil dirinya sendiri secara berulang untuk menyelesaikan tugas tertentu. Contoh fungsi rekursif yang paling sederhana adalah faktorial, yang dapat didefinisikan sebagai berikut:



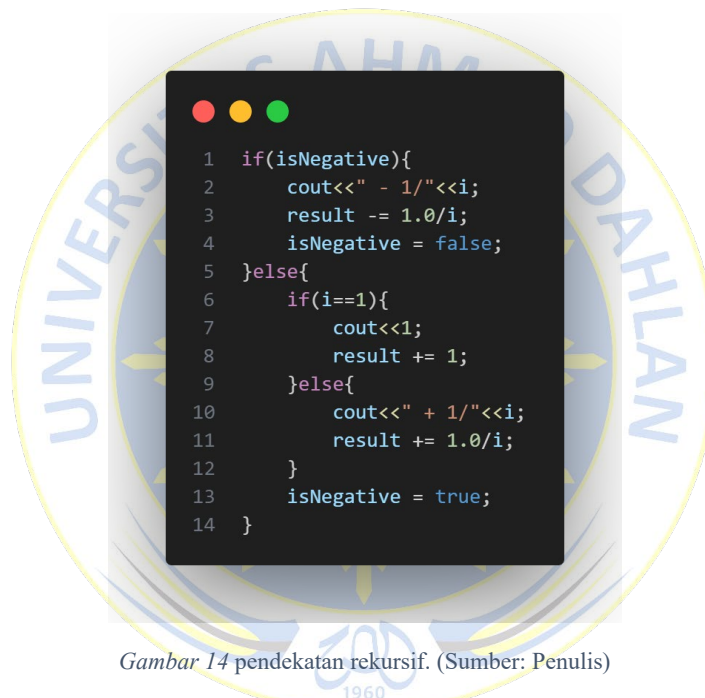
```
1 int factorial(int n){
2     if(n == 0 || n == 1){
3         return 1;
4     }else{
5         return n * factorial(n-1);
6     }
}
```

Gambar 13 Penggunaan fungsi rekursif seharusnya. (Sumber: Penulis)

Namun, program ini sebenarnya tidak sepenuhnya menggunakan fungsi rekursif. Meskipun terdapat perulangan pada bagian `series`, perulangan tersebut dilakukan secara manual dengan menggunakan statement `goto`. Artinya, program ini masih menggunakan paradigma pemrograman imperatif atau prosedural yang mengandalkan serangkaian instruksi untuk menyelesaikan tugas tertentu.

Selain itu, apabila kita ingin menulis program $F(n)$ dengan metode rekursif, maka kita akan memerlukan sebuah fungsi rekursif yang melakukan penghitungan nilai setiap elemen dalam deret dan kemudian menjumlahkan semua nilai tersebut. Namun, pada implementasi program ini, kita tidak menemukan adanya fungsi rekursif tersebut. Sebaliknya, perhitungan dilakukan dalam sebuah loop menggunakan statement while dan goto.

Meskipun demikian, ada beberapa hal di dalam program ini yang dapat dikatakan memiliki sifat rekursif. Secara umum, sifat rekursif adalah kemampuan suatu algoritma atau program untuk memecah masalah yang kompleks menjadi submasalah yang lebih kecil dan dapat diselesaikan dengan cara yang sama. Pada program ini, kita dapat melihat bahwa saat menghitung nilai $F(n)$, kita bisa membagi perhitungan menjadi dua bagian, yaitu penghitungan elemen positif dan negatif. Hal ini dapat dilihat pada bagian kode berikut:



Gambar 14 pendekatan rekursif. (Sumber: Penulis)

Dalam bagian ini, kita dapat menganggap bahwa penghitungan elemen positif dan negatif sebagai submasalah yang masing-masing dapat diselesaikan secara rekursif, yaitu dengan melakukan pemanggilan fungsi yang sama di dalam dirinya sendiri. Meskipun kodingan ini tidak menggunakan struktur rekursif secara eksplisit, namun setiap elemen pada deret tersebut bisa dilihat sebagai sub-problem dari masalah yang lebih besar.

Namun, perlu diketahui bahwa pendekatan rekursif bukanlah satu-satunya cara untuk menyelesaikan masalah. Pemilihan pendekatan tergantung pada sifat masalah yang ingin diselesaikan, serta preferensi atau kebiasaan programmer. Selain itu, penggunaan teknik lain seperti iterasi atau pemrograman dinamis juga dapat memberikan solusi yang baik dalam beberapa kasus.

Konversi Nomor 2:

```

1  #include <iostream>
2  using namespace std;
3
4  double series(int n){
5      if(n==1){
6          return 1;
7      }else{
8          if(n%2==0){
9              return -1.0/n + series(n-1);
10             }else{
11                 return 1.0/n + series(n-1);
12             }
13         }
14     }
15
16     void printFormula(int n){
17         if(n==1){
18             cout<<"1";
19         }else{
20             printFormula(n-1);
21             if(n%2==0){
22                 cout<<" - 1/"<<n;
23             }else{
24                 cout<<" + 1/"<<n;
25             }
26         }
27     }
28
29     int main(){
30         int number;
31         cout<<"Masukkan nilai n: "; cin>>number;
32         cout<<"F("<<number<<") = "; printFormula(number);
33         cout<<"\nHasil: "<<series(number)<<endl;
34     }

```

Gambar 15 Kodingan menggunakan subprogram untuk mencari deret. (Sumber: Penulis)

Berikut adalah outputnya:

```

1  #include <iostream>
2  using namespace std;
3
4  double series(int n){
5      if(n==1){
6          return 1;
7      }else{
8          if(n%2==0){
9              return -1.0/n + series(n-1);
10             }else{
11                 return 1.0/n + series(n-1);
12             }
13         }
14     }
15
16     void printFormula(int n){
17         if(n==1){
18             cout<<"1";
19         }else{
20             printFormula(n-1);
21             if(n%2==0){
22                 cout<<" - 1/"<<n;
23             }else{
24                 cout<<" + 1/"<<n;
25             }
26         }
27     }
28
29     int main(){
30         int number;
31         cout<<"Masukkan nilai n: "; cin>>number;
32         cout<<"F("<<number<<") = "; printFormula(number);
33         cout<<"\nHasil: "<<series(number)<<endl;
34     }

```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```

PS D:\Document Ndik\Kuliah\Semester 2\Algoritma Pemrograman\Praktikum\Pertemuan 5> cd "d:\Document Ndik\Kuliah\Semester 2\Algoritma Pemrograman\Praktikum\Pertemuan 5"; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan nilai n: 5
F(5) = 1 - 1/2 + 1/3 - 1/4 + 1/5
Hasil: 0.78333
PS D:\Document Ndik\Kuliah\Semester 2\Algoritma Pemrograman\Praktikum\Pertemuan 5>

```

Gambar 16 Output series(5). (Sumber: Penulis)

Kodingan tersebut merupakan sebuah program C++ yang menghitung nilai dari sebuah deret dan menampilkan rumus serta hasilnya ke layar.

Pada bagian awal program, kita sudah dapat melihat bahwa terdapat sebuah fungsi bernama "series" yang menerima parameter n . Fungsi ini berfungsi untuk menghitung nilai dari deret dengan bilangan bulat positif n sebagai jumlah suku dalam deret tersebut.

Fungsi series menggunakan pendekatan rekursif, dimana pada kasus dasar (base case) ketika n bernilai 1, fungsi akan langsung mengembalikan nilai 1. Sedangkan pada kasus umum, jika n genap, maka fungsi akan mengembalikan nilai $-1/n$ ditambah dengan hasil fungsi $\text{series}(n-1)$ yang dipanggil secara rekursif. Namun jika n ganjil, fungsi akan mengembalikan nilai $1/n$ ditambah dengan hasil fungsi $\text{series}(n-1)$ yang juga dipanggil secara rekursif.

Dalam kodingan ini, terdapat pula fungsi "printFormula" yang bertujuan untuk menampilkan rumus deret ke layar. Fungsi ini juga menggunakan pendekatan rekursif, dimulai dari kasus dasar ketika n sama dengan 1, maka akan menampilkan nilai 1. Pada kasus umum, fungsi akan memanggil dirinya sendiri dengan parameter $n-1$, kemudian mengecek apakah n genap atau ganjil, dan menampilkan rumus sesuai dengan kondisi tersebut. Jika n genap, maka akan menampilkan tanda minus dan pecahan $1/n$, sedangkan jika n ganjil maka akan menampilkan tanda plus dan pecahan $1/n$.

Pada bagian akhir program, terdapat fungsi "main" yang digunakan sebagai entry point dari program. Fungsi ini meminta input dari user berupa bilangan bulat positif n . Kemudian, program akan menampilkan rumus deret dengan memanggil fungsi "printFormula" dengan parameter n , dan kemudian menampilkan hasil deret dengan memanggil fungsi "series" dengan parameter n dan menampilkannya ke layar.

Dalam kodingan ini, fungsi "series" dan "printFormula" menggunakan pendekatan rekursif untuk menghitung nilai deret serta menampilkan rumusnya secara urut ke layar.

Pertama-tama, fungsi series akan mengecek apakah n sudah mencapai kasus dasar ($n=1$). Jika ya, maka fungsi akan langsung mengembalikan nilai 1. Namun jika tidak, fungsi akan mengecek apakah n genap atau ganjil. Jika n genap, maka fungsi akan mengembalikan nilai $-1/n$ ditambah dengan hasil pemanggilan fungsi series dengan parameter $n-1$. Sedangkan jika n ganjil, fungsi akan mengembalikan nilai $1/n$ ditambah dengan hasil pemanggilan fungsi series dengan parameter $n-1$.

Selanjutnya, fungsi printFormula juga menggunakan pendekatan rekursif, dimulai dari kasus dasar ketika n sama dengan 1, maka akan menampilkan nilai 1. Pada kasus umum, fungsi akan memanggil dirinya sendiri dengan parameter $n-1$, kemudian mengecek apakah n genap atau ganjil, dan menampilkan rumus sesuai dengan kondisi tersebut. Jika n genap, maka akan menampilkan tanda minus dan pecahan $1/n$, sedangkan jika n ganjil maka akan menampilkan tanda plus dan pecahan $1/n$.

Untuk mengakses kodingan, dapat melihat link berikut:

<https://github.com/IRedDragonICY/Programming-Algorithms>