

JOBSHEET IV

BRUTE FORCE DAN DIVIDE CONQUER

4.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mahasiswa mampu membuat algoritma brute force dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algoritma brute force dan divide-conquer


4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

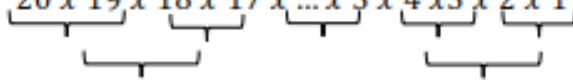
Faktorial
nilai: int
faktorialBF(): int
faktorialDC(): int

Berdasarkan diagram class di atas, akan dibuat program class dalam Java. Untuk menghitung nilai faktorial suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Jika digambarkan terdapat perbedaan proses perhitungan 2 jenis algoritma tersebut sebagai berikut :

Tahapan pencarian nilai faktorial dengan algoritma Brute Force :

$$20! = 20 \times 19 \times 18 \times 17 \times \dots \times 5 \times 4 \times 3 \times 2 \times 1$$


Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer :

$$20! = 20 \times 19 \times 18 \times 17 \times \dots \times 5 \times 4 \times 3 \times 2 \times 1$$


4.2.1 Langkah-langkah Percobaan

1. Buat Project baru, dengan nama "BruteForceDivideConquer". Buat package dengan nama minggu5.
2. Buatlah class baru dengan nama **Faktorial**
3. Lengkapi class **Faktorial** dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
 - a) Tambahkan atribut nilai

```
public int nilai;
```

- b) Tambahkan method faktorialBF() nilai

```
public int faktorialBF(int n){
    int fakto = 1;
    for (int i = 1; i <= n; i++) {
        fakto = fakto * i;
    }
    return fakto;
}
```

- c) Tambahkan method faktorialDC() nilai

```
public int faktorialDC(int n){
    if (n==1) {
        return 1;
    }
    else
    {
        int fakto = n * faktorialDC(n-1);
        return fakto;
    }
}
```

4. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.

- a) Di dalam fungsi main sediakan komunikasi dengan user untuk menginputkan jumlah angka yang akan dicari nilai faktorialnya

```
Scanner sc = new Scanner(System.in);
System.out.println("=====");
System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = sc.nextInt();
```

- b) Buat Array of Objek pada fungsi main, kemudian inputkan beberapa nilai yang akan dihitung faktorialnya

```
Faktorial [] fk = new Faktorial[elemen];
for (int i = 0; i < elemen; i++) {
    fk[i] = new Faktorial();
    System.out.print("Masukkan nilai data ke-"+(i+1)+" : ");
    fk[i].nilai = sc.nextInt();
}
```

- c) Tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()

```

System.out.println("=====");
System.out.println("Hasil Faktorial dengan Brute Force");
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialBF(fk[i].nilai));
}
System.out.println("=====");
System.out.println("Hasil Faktorial dengan Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialDC(fk[i].nilai));
}
System.out.println("=====");
    
```

d) Pastikan program sudah berjalan dengan baik!

4.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```

run:
=====
Masukkan jumlah elemen yang ingin dihitung : 3
Masukkan nilai data ke-1 : 5
Masukkan nilai data ke-2 : 8
Masukkan nilai data ke-3 : 3
=====
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====
Hasil Faktorial dengan Divide and Conquer
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====
BUILD SUCCESSFUL (total time: 7 seconds)
    
```

4.2.3 Pertanyaan

1. Jelaskan mengenai base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial! pencarian nilai faktorial, algoritma ini memiliki pendekatan dasar yang dimulai dengan mengidentifikasi kondisi dasar (base case) di mana faktorial dari 0 atau 1 adalah 1, sehingga berfungsi sebagai titik akhir rekursi. Selanjutnya, langkah-langkahnya melibatkan pembagian masalah menjadi submasalah yang lebih kecil, yaitu membagi nilai yang dicari menjadi dua. Setelah itu, untuk setiap submasalah, faktorial dari nilai yang lebih kecil dihitung secara terpisah. Terakhir, solusi-solusi dari submasalah-submasalah tersebut digabungkan dengan mengalikannya untuk mendapatkan hasil akhir. Dengan menggunakan pendekatan Divide and Conquer, kompleksitas waktu dari pencarian nilai faktorial dapat diminimalkan, menghasilkan kinerja yang lebih efisien dan cepat dalam menyelesaikan masalah.

2. Pada implementasi Algoritma Divide and Conquer Faktorial apakah lengkap terdiri dari 3 tahapan divide, conquer, combine? Jelaskan masing-masing bagiannya pada kode program!

Divide:

Bagi masalah menjadi submasalah yang lebih kecil.

Dalam pencarian faktorial, kita dapat membagi masalah menjadi dua bagian dengan cara membagi nilai yang dicari menjadi dua.

Conquer:

Selesaikan submasalah secara terpisah.

Dalam hal ini, untuk setiap submasalah, kita akan mencari faktorial dari nilai yang lebih kecil.

Combine:

Gabungkan solusi-solusi dari submasalah-submasalah tersebut untuk mendapatkan solusi akhir.

Dalam kasus pencarian faktorial, kita akan mengalikan solusi-solusi submasalah untuk mendapatkan hasil akhir.

3. Apakah memungkinkan perulangan pada method `faktorialBF()` dirubah selain menggunakan `for`?Buktikan!

```

1  class faktorial {
2      int nilai;
3
4      // Metode untuk menghitung faktorial dengan rekursi (Brute Force)
5      int faktorialBF(int n) {
6          if (n == 0 || n == 1) {
7              return 1;
8          } else {
9              return n * faktorialBF(n - 1);
10         }
11     }
12 }
```

4. Tambahkan pengecekan waktu eksekusi kedua jenis method tersebut!

```

1 System.out.println("=====");
2 System.out.println("Hasil Faktorial dengan Brute force");
3 for (int i = 0; i < elemen; i++) {
4     long startTime = System.currentTimeMillis();
5     int resultBF = fk[i].faktorialBF(fk[i].nilai);
6     long endTime = System.currentTimeMillis();
7     System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah : " + resultBF + ", Waktu eksekusi: " + (endTime - startTime) + " ms");
8 }
9
10 System.out.println("=====");
11 System.out.println("Hasil Faktorial dengan Divide and Conquer");
12 for (int i = 0; i < elemen; i++) {
13     long startTime = System.currentTimeMillis();
14     int resultDC = fk[i].faktorialDC(fk[i].nilai);
15     long endTime = System.currentTimeMillis();
16     System.out.println("Faktorial dari nilai " + fk[i].nilai + " adalah : " + resultDC + ", Waktu eksekusi: " + (endTime - startTime) + " ms");
17 }
    
```

5. Buktikan dengan inputan elemen yang di atas 20 angka, apakah ada perbedaan waktu eksekusi?

```

Masukan nilai data ke - 20 : 8
Masukan nilai data ke - 21 : 3

=====
Hasil Faktorial dengan Brute force
Faktorial dari nilai 5 adalah : 120, Waktu eksekusi: 0 ms
Faktorial dari nilai 8 adalah : 40320, Waktu eksekusi: 0 ms
Faktorial dari nilai 3 adalah : 6, Waktu eksekusi: 0 ms
Faktorial dari nilai 5 adalah : 120, Waktu eksekusi: 0 ms
Faktorial dari nilai 8 adalah : 40320, Waktu eksekusi: 0 ms
Faktorial dari nilai 3 adalah : 6, Waktu eksekusi: 0 ms
Faktorial dari nilai 5 adalah : 120, Waktu eksekusi: 0 ms
Faktorial dari nilai 8 adalah : 40320, Waktu eksekusi: 0 ms
Faktorial dari nilai 3 adalah : 6, Waktu eksekusi: 0 ms
Faktorial dari nilai 5 adalah : 120, Waktu eksekusi: 0 ms
    
```

4.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

Pada praktikum ini kita akan membuat program class dalam Java. Untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer.

4.3.1 Langkah-langkah Percobaan

1. Di dalam paket `minggu5`, buatlah class baru dengan nama `Pangkat`. Dan di dalam class `Pangkat` tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya

```
public int nilai,pangkat;
```

2. Pada class `Pangkat` tersebut, tambahkan method `PangkatBF()`

```
public int pangkatBF(int a,int n){
    int hasil=1;
    for (int i = 0; i < n; i++) {
        hasil = hasil * a;
    }
    return hasil;
}
```

3. Pada class Pangkat juga tambahkan method PangkatDC ()

```
public int pangkatDC(int a,int n){
    if (n==0) {
        return 1;
    }
    else
    {
        if(n%2==1)//bilangan ganjil
            return (pangkatDC(a,n/2)*pangkatDC(a,n/2)*a);
        else//bilangan genap
            return (pangkatDC(a,n/2)*pangkatDC(a,n/2));
    }
}
```

4. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
5. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah nilai yang akan dihitung pangkatnya.

```
Scanner sc = new Scanner(System.in);
System.out.println("=====");
System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = sc.nextInt();
```

6. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.

```
Pangkat [] png = new Pangkat[elemen];

for (int i = 0; i < elemen; i++) {
    png[i] = new Pangkat();
    System.out.print("Masukkan nilai yang akan dipangkatkan ke-"+(i+1)+" : ");
    png[i].nilai = sc.nextInt();
    System.out.print("Masukkan nilai pemangkat ke-"+(i+1)+" : ");
    png[i].pangkat = sc.nextInt();
}
```

7. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF () dan PangkatDC () .

```

System.out.println("=====");
System.out.println("Hasil Pangkat dengan Brute Force");
for (int i = 0; i < elemen; i++) {
    System.out.println("Nilai "+png[i].nilai+" pangkat "+png[i].pangkat+" adalah : "+png[i].pangkatBF(png[i].nilai,png[i].pangkat));
}
System.out.println("=====");
System.out.println("Hasil Pangkat dengan Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    System.out.println("Nilai "+png[i].nilai+" pangkat "+png[i].pangkat+" adalah : "+png[i].pangkatDC(png[i].nilai,png[i].pangkat));
}
System.out.println("=====");
    
```

4.3.2 Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.

```

run:
=====
Masukkan jumlah elemen yang ingin dihitung : 2
Masukkan nilai yang akan dipangkatkan ke-1 : 6
Masukkan nilai pemangkat ke-1 : 2
Masukkan nilai yang akan dipangkatkan ke-2 : 4
Masukkan nilai pemangkat ke-2 : 3
=====
Hasil Pangkat dengan Brute Force
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
=====
Hasil Pangkat dengan Divide and Conquer
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
=====
BUILD SUCCESSFUL (total time: 10 seconds)
    
```

4.3.3 Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu `PangkatBF()` dan `PangkatDC()` !

Pangkat Brute Force (`pangkatBF()`): Metode ini menggunakan pendekatan langsung dengan mengalikan nilai yang akan dipangkatkan sebanyak pangkat yang diinginkan. Artinya, operasi perkalian dilakukan sebanyak n kali, di mana n adalah pangkat yang diinginkan.

Pangkat Divide and Conquer (`pangkatDC()`): Metode ini menggunakan pendekatan rekursif untuk menghitung pangkat. Ketika menghitung pangkat dari suatu angka, metode ini membagi pangkat menjadi dua bagian yang lebih kecil (divide) dan kemudian menggabungkan (conquer) hasilnya. Dalam hal ini, algoritma memanfaatkan sifat eksponensial untuk mengurangi jumlah operasi yang dibutuhkan.

2. Pada method `PangkatDC()` terdapat potongan program sebagai berikut:

```

if(n%2==1)//bilangan ganjil
    return (pangkatDC(a,n/2)*pangkatDC(a,n/2)*a);
else//bilangan genap
    return (pangkatDC(a,n/2)*pangkatDC(a,n/2));
    
```

Jelaskan arti potongan kode tersebut

Potongan kode if ($n \% 2 == 1$) pada metode pangkatDC() menentukan apakah pangkat (n) adalah bilangan ganjil atau genap. Jika pangkat adalah bilangan ganjil, maka algoritma akan membagi pangkat menjadi dua bagian yang sama, lalu mengalikan hasilnya dengan nilai yang akan dipangkatkan. Sedangkan jika pangkat adalah bilangan genap, algoritma hanya membagi pangkat menjadi dua bagian yang sama tanpa mengalikan dengan nilai yang akan dipangkatkan.

3. Apakah tahap *combine* sudah termasuk dalam kode tersebut? Tunjukkan!

Tahap "combine" dalam algoritma Divide and Conquer terjadi ketika hasil perhitungan pada setiap subdivisi (bagian yang lebih kecil) digabungkan kembali menjadi hasil akhir. Dalam kasus kode ini, tahap "combine" terjadi saat hasil perhitungan pada setiap subdivisi digunakan untuk mengalikan atau menggabungkan kembali hasilnya.

4. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.

```

1  class pangkat {
2      public int nilai, pangkat;
3
4      public pangkat(int nilai, int pangkat) {
5          this.nilai = nilai;
6          this.pangkat = pangkat;
7      }
8
9      public int pangkatBF(int a, int n){
10         int hasil = 1;
11         for (int i = 0; i < n; i++) {
12             hasil = hasil * a;
13         }
14         return hasil;
15     }
16
17     public int pangkatDC (int a, int n){
18         if (n == 0) {
19             return 1;
20         } else {
21             if (n % 2 == 1)
22                 return (pangkatDC(a,n/2) * pangkatDC(a, n/2) * a);
23             else
24                 return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
25         }
26     }
27 }
    
```

5. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!

Menu telah ditambahkan untuk memilih metode perhitungan pangkat yang akan dijalankan. Menu memilih antara metode Brute Force atau Divide and Conquer sesuai dengan pilihan yang Anda masukkan.

4.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses *divide*, *conquer*, dan *combine* diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

4.4.1 Langkah-langkah Percobaan

1. Pada paket minggu5. Buat class baru yaitu class `Sum`. Di dalam class tersebut terdapat beberapa atribut jumlah elemen array, array, dan juga total. Tambahkan pula konstruktor pada class `Sum`.

```
public int elemen;
public double keuntungan[];
public double total;
```

```
Sum(int elemen){
    this.elemen = elemen;
    this.keuntungan=new double[elemen];
    this.total = 0;
}
```

2. Tambahkan method `TotalBF()` yang akan menghitung total nilai array dengan cara *iterative*.

```
double totalBF(double arr[]){
    for (int i = 0; i < elemen; i++) {
        total = total + arr[i];
    }
    return total;
}
```

3. Tambahkan pula method `TotalDC()` untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer

```
double totalDC(double arr[], int l, int r){
    if(l==r)
        return arr[l];
    else if(l<r){
        int mid=(l+r)/2;
        double lsum=totalDC(arr,l,mid-1);
        double rsum=totalDC(arr,mid+1,r);
        return lsum+rsum+arr[mid];
    }

    return 0;
}
```

4. Buat class baru yaitu `MainSum`. Di dalam kelas ini terdapat method `main`. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class `Sum`

```
Scanner sc = new Scanner(System.in);
System.out.println("=====");
System.out.println("Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)");
System.out.print("Masukkan jumlah bulan : ");
int elm = sc.nextInt();
```

5. Karena yang akan dihitung adalah total nilai keuntungan, maka ditambahkan pula pada method `main` mana array yang akan dihitung. Array tersebut merupakan atribut yang terdapat di class `Sum`, maka dari itu dibutuhkan pembuatan objek `Sum` terlebih dahulu.

```
Sum sm = new Sum(elm);
System.out.println("=====");
for (int i = 0; i < sm.elemen; i++) {
    System.out.print("Masukkan untung bulan ke - "+(i+1)+" = ");
    sm.keuntungan[i] = sc.nextDouble();
}
```

6. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)

```
System.out.println("=====");
System.out.println("Algoritma Brute Force");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = "+sm.totalBF(sm.keuntungan));
System.out.println("=====");
System.out.println("Algoritma Divide Conquer");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = "+sm.totalDC(sm.keuntungan, 0, sm.elemen-1));
```

4.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
run:
=====
Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)
Masukkan jumlah bulan : 5
=====
Masukkan untung bulan ke - 1 = 8.5
Masukkan untung bulan ke - 2 = 9.54
Masukkan untung bulan ke - 3 = 7.2
Masukkan untung bulan ke - 4 = 9.1
Masukkan untung bulan ke - 5 = 6
=====
Algoritma Brute Force
Total keuntungan perusahaan selama 5 bulan adalah = 40.339999999999996
=====
Algoritma Divide Conquer
Total keuntungan perusahaan selama 5 bulan adalah = 40.34
BUILD SUCCESSFUL (total time: 11 seconds)
```



4.4.3 Pertanyaan

1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method `TotalBF()` ataupun `TotalDC()`

`TotalBF()`: Metode Brute Force melakukan penjumlahan seluruh keuntungan secara berurutan dari bulan pertama hingga bulan terakhir.

`TotalDC()`: Metode Divide and Conquer membagi masalah menjadi submasalah yang lebih kecil, kemudian menggabungkan solusi submasalah tersebut. Di sini, submasalah adalah membagi array menjadi dua bagian, kemudian menghitung total keuntungan dari masing-masing bagian dan menggabungkannya.

2. Perhatikan output dari kedua jenis algoritma tersebut bisa jadi memiliki hasil berbeda di belakang koma. Bagaimana membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma tersebut.

Untuk membatasi output di belakang koma, Anda dapat menggunakan fungsi `String.format()` dengan format desimal yang diinginkan, misalnya `String.format("%.2f", totalKeuntungan)` akan membatasi hasil menjadi dua angka di belakang koma.

3. Mengapa terdapat formulasi *return value* berikut?Jelaskan!

```
return lsum+rsum+arr[mid];
```

Formulasi Return Value `return lsum + rsum + arr[mid]`:

Di sini, `lsum` adalah total keuntungan dari subarray kiri, `rsum` adalah total keuntungan dari subarray kanan, dan `arr[mid]` adalah keuntungan bulan tengah. Jadi, formulasi ini mengembalikan total keuntungan dari seluruh array, yang terdiri dari total keuntungan dari subarray kiri, subarray kanan, dan keuntungan bulan tengah.

4. Kenapa dibutuhkan variable `mid` pada method `TotalDC()`?

Kebutuhan Variabel `mid` pada `TotalDC()`:

Variabel `mid` digunakan untuk menandai indeks tengah dari array. Ini diperlukan untuk membagi array menjadi dua bagian dalam metode Divide and Conquer.

5. Program perhitungan keuntungan suatu perusahaan ini hanya untuk satu perusahaan saja. Bagaimana cara menghitung sekaligus keuntungan beberapa bulan untuk beberapa perusahaan.(Setiap perusahaan bisa saja memiliki jumlah bulan berbeda-beda)? Buktikan dengan program!

```

1 import java.util.Scanner;
2
3 public class mainsum {
4
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         System.out.println("=====");
8         System.out.println("Program menghitung keuntungan total (satuan juta, misal 5.9)");
9         System.out.print("Masukan jumlah bulan : ");
10        int elm = sc.nextInt();
11
12        sum sm = new sum(elm);
13        System.out.println("=====");
14        for (int i = 0; i < sm.elemen; i++) {
15            System.out.print("Masukan untung bulan ke = " + (i + 1) + " = ");
16            sm.keuntungan[i] = sc.nextDouble();
17        }
18
19        System.out.println("=====");
20        System.out.println("Algoritma Brute Force");
21        System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = " + sm.totalBF(sm.keuntungan));
22        System.out.println("=====");
23        System.out.println("Algoritma Devide Conquer");
24        System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = " + sm.totalDC(sm.keuntungan, 0, sm.elemen - 1));
25
26        sc.close();
27    }
28 }

```

```

1 class Sum {
2     public int elemen;
3     public double keuntungan[];
4     public double total;
5
6     Sum(int elemen){
7         this.elemen = elemen;
8         this.keuntungan = new double[elemen];
9         this.total = 0;
10    }
11
12    double totalBF(double arr[]){
13        for (int i = 0; i < elemen; i++) {
14            total = total + arr[i];
15        }
16        return total;
17    }
18
19    double totalDC(double arr[], int L, int r){
20        if (L == r) {
21            return arr[L];
22        } else if (L < r) {
23            int mid = (L + r) / 2;
24            double lsum = totalDC(arr, L, mid);
25            double rsum = totalDC(arr, mid + 1, r);
26            return lsum + rsum + arr[mid];
27        }
28        return 0;
29    }
30 }

```



4.5 Latihan Praktikum

Buatlah kode program untuk menghitung nilai akar dari suatu bilangan dengan algoritma Brute Force dan Divide Conquer! *Jika bilangan tersebut bukan merupakan kuadrat sempurna, bulatkan angka ke bawah.*