

LAPORAN PRAKTIKUM

“PRAKTIKUM 6: BILANGAN BULAT”

Diajukan untuk memenuhi salah satu praktikum Mata Kuliah Matematika Diskrit yang di
ampu oleh:

Nur Rochmah Dyah PA, S.T., M.Kom



Disusun Oleh:

Mohammad Farid Hendianto 2200018401

Selasa 12.00-13.30

PROGRAM STUDI INFORMATIKA
UNIVERSITAS AHMAD DAHLAN
FAKULTAS TEKNOLOGI INDUSTRI
TAHUN 2023

Berikut adalah kodingan dari modul yang belum dibenarkan:

```

1 #include <iostream>
2 Using namespace std;
3
4 main()
5 {
6     clrscr();
7     int a,b,c,d;
8     int p;
9     int faktor1,faktor2,kpk,fpb;
10
11     cout<<"Masukan Pilihan anda ?\n";
12     cout<<"1. Menentukan KPK\n";
13     cout<<"2. Menentukan FPB\n";
14     cout<<"3. Exit\n";
15     cin>>p;
16
17     switch (p){
18     case 1:
19         cout<<"Menghitung KPK\n";
20         cout<<"Masukan Bilangan Pertama : \n";
21         cin>>a;
22         cout<<"Masukan Bilangan Kedua : \n";
23         cin>>b;
24         if (a>b)
25             if (a%b){
26                 for(c=0;c<=a;c++){
27                     if(a%c){ //lanjutkan
28                         faktor1=c;
29                     }
30                     for(d=0;d<=b;d++){
31                         b%d;
32                         if (b%d);
33                         //lanjutkan
34                         else
35                             faktor2=d;
36                     }
37                 }
38                 kpk=a;
39             }
40             else if (b%a){
41                 for(d=0;d<=a;d++){
42                     if(b%d);
43                     //lanjutkan
44                     else
45                         faktor1=d;
46                 }
47                 for(c=0;c<=b;c++){
48                     if (a%c);
49                     //lanjutkan
50                     else
51                         faktor2=c;
52                 }
53             }
54             else
55                 kpk=b;
56             fpb=faktor1*faktor2;
57             cout<<"Bilangan pertama : "<<endl;
58             cout<<"Bilangan kedua : "<<endl;
59             cout<<"KPK : "<<
60             break;
61             break;
62     case 2:
63         cout<<"Menghitung FPB\n";
64         cout<<"Masukan Bilangan pertama : \n";
65         cin>>a;
66         cout<<"Masukan bilangan kedua : \n";
67         cin>>b;
68         if (a
69         if (b%a){
70             for(c=0;c>=a;c--){
71                 if(c%a){ //lanjutkan
72                     else
73                         faktor1=c;
74                 }
75                 for(d=0;d>=b;d--){
76                     b%d;
77                     if (d%b){ //lanjutkan
78                         else faktor2=d;
79                 }
80                 for(c=0;c>=b;c--){
81                     if (c%a){ //lanjutkan
82                         else
83                             \faktor1=c;
84                 }
85             }
86             else
87                 fpb=b;
88             fpb=faktor1*faktor2;
89             cout<<"Bilangan pertama : "<<
90             cout<<"Bilangan kedua : "<<
91             cout<<"FPB : "<<
92             break;
93             case 3:
94                 cout<<"Exit Now !!!\n";
95                 break;
96             default:
97                 cout<<"Error !!!\n";
98                 getch();
99                 return 0;
100 }

```

Gambar 1 Kodingan modul yang masih banyak error. (Sumber: Penulis)

Berikut adalah kodingan yang sudah di benarkan, dan lebih efisien:

```

1 #include <iostream>
2
3 using namespace std;
4
5 class Math {
6 public:
7     // Algoritma Brute Force
8     int getKPK(int num1, int num2) {
9         int result = 0;
10        for (int i = 1; i <= num1 * num2; i++) {
11            if (i % num1 == 0 && i % num2 == 0) {
12                result = i;
13                break;
14            }
15        }
16        return result;
17    }
18    int getFPB(int num1, int num2) {
19        int result = 0;
20        for (int i = 1; i <= num1 && i <= num2; i++) {
21            if (num1 % i == 0 && num2 % i == 0) {
22                result = i;
23            }
24        }
25        return result;
26    }
27    // Algoritma Euclidean
28    int getFPB(int num1, int num2) {
29        int fpb = getFPB(num1, num2);
30        return (num1 * num2) / fpb;
31    }
32
33    // int getFPB(int num1, int num2) {
34    //     while (num2 != 0) {
35    //         int temp = num2;
36    //         num2 = num1 % num2;
37    //         num1 = temp;
38    //     }
39    //     return num1;
40    // }
41 };
42
43 class Menu {
44 public:
45     int displayMenu() {
46         int choice;
47         cout << "Masukkan pilihan anda:" << endl;
48         << "1. Menentukan KPK" << endl;
49         << "2. Menentukan FPB" << endl;
50         << "3. Keluar" << endl;
51         << "> ";
52         cin >> choice;
53         return choice;
54     }
55     void inputNum(int& num1, int& num2) {
56         cout << "Masukkan bilangan pertama: ";
57         cin >> num1;
58         cout << "Masukkan bilangan kedua: ";
59         cin >> num2;
60     }
61     void pressEnterToContinue() {
62         cout << "Tekan Enter untuk melanjutkan...";
63         cin.ignore();
64         cin.get();
65     }
66 };
67
68 class Program {
69 public:
70     void run() {
71         Math math;
72         Menu menu;
73         int choice;
74         int num1, num2;
75
76         while (true) {
77             system("cls");
78             choice = menu.displayMenu();
79
80             system("cls");
81             switch (choice) {
82                 case 1:
83                 << "Menentukan KPK" << endl;
84                 menu.inputNum(num1, num2);
85
86                 cout << "Bilangan pertama: " << num1 << endl;
87                 cout << "Bilangan kedua: " << num2 << endl;
88                 cout << "KPK: " << math.getKPK(num1, num2) << endl;
89                 break;
90
91                 case 2:
92                 << "Menentukan FPB" << endl;
93                 menu.inputNum(num1, num2);
94
95                 cout << "Bilangan pertama: " << num1 << endl;
96                 cout << "Bilangan kedua: " << num2 << endl;
97                 cout << "FPB: " << math.getFPB(num1, num2) << endl;
98                 break;
99
100                case 3:
101                << "Keluar dari program..." << endl;
102                return;
103
104                default:
105                << "Pilihan tidak valid!" << endl;
106            }
107            menu.pressEnterToContinue();
108        }
109    }
110 };
111
112 int main() {
113     Program program;
114     program.run();
115     return 0;
116 }

```

Gambar 2 Source code untuk mencari KPK dan FPB. (Sumber: Penulis)

Kodingan pertama yang terdapat pada modul memiliki beberapa kesalahan yang mungkin diakibatkan oleh ketidaktepatan saat pengetikan kode. Dalam menjalankan program, kodingan akan mencetak pilihan menu untuk menentukan KPK atau FPB dari dua bilangan bulat positif dan bisa keluar dari program. Penjelasan lebih rinci tentang kodingan pertama dan kodingan kedua dapat dilihat di bawah ini.

Pada kodingan pertama, terdapat beberapa kesalahan sintaksis. Pertama-tama, menggunakan "Using namespace std" tidak disarankan, karena dapat menyebabkan masalah dengan namespace yang tidak sengaja ditimpa oleh kode lain dalam program kita. Lebih baik menggunakan std:: sebelum menggunakan fungsi standar C++ seperti cout dan cin. Selanjutnya, fungsi clrscr() yang terdapat pada baris ke-4 tidak didefinisikan sebelumnya, sehingga menyebabkan kesalahan saat memanggil fungsi tersebut. Kemudian, variabel p yang digunakan untuk menyimpan pilihan user harus dideklarasikan terlebih dahulu. Di samping itu, variabel faktor1, faktor2, kpk, dan fpb tidak perlu dideklarasikan di awal, karena nilainya belum diketahui. Hal ini dapat menghemat ruang penyimpanan dan membuat kode lebih mudah dibaca.

Pada bagian switch-case, ada banyak blok if-else berlebihan yang dapat mengganggu pembacaan kode. Selain itu, logika matematika yang digunakan pada percabangan if-else juga salah, sehingga perhitungan KPK dan FPB pada kode tidak benar. Saat mencari faktor bilangan, perulangan for harus dimulai dari 1, bukan 0, karena bilangan yang habis dibagi oleh 0 adalah tidak terdefinisi dan dapat menyebabkan kesalahan program.

Kodingan kedua membungkus beberapa fungsi ke dalam kelas untuk meningkatkan struktur kode dan mengurangi penggunaan variabel global. Fungsi getKPK() dan getFPB() didefinisikan di dalam kelas Math dengan algoritma Brute Force dan Euclidean masing-masing untuk menentukan KPK dan FPB dua bilangan bulat positif. Algoritma Brute Force menggunakan perulangan untuk mencari bilangan terkecil yang dapat dibagi oleh kedua bilangan, sedangkan algoritma Euclidean menggunakan sisa hasil bagi antara dua bilangan untuk menemukan FPB, dan kemudian KPK dapat dihitung dengan mudah dari FPB.

Kelas Menu berfungsi untuk menampilkan menu pilihan pada layar dan meminta user untuk memasukkan dua bilangan yang akan diproses. Kemudian, Program class menjalankan loop while yang tak henti-hentinya sampai user memilih opsi "Keluar" dari menu. Program Class kemudian memanggil fungsi yang sesuai dari Math dan Menu kelas untuk mengeksekusi program, dan menambah fitur "Tekan Enter untuk melanjutkan..." untuk membuat program lebih interaktif.

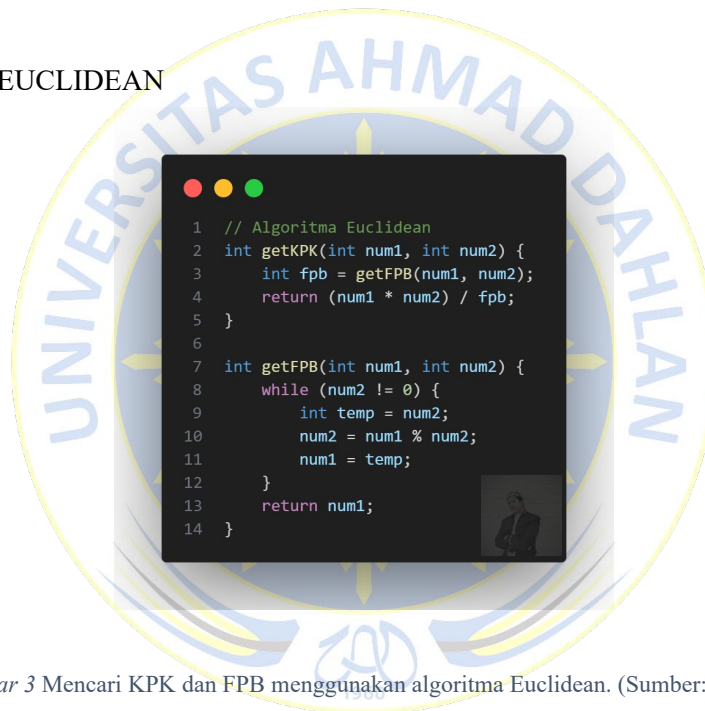
Kodingan pertama yang terdapat pada modul memiliki beberapa kesalahan sintaksis, sehingga program yang dihasilkan tidak berjalan dengan baik. Kodingan tersebut merupakan contoh sederhana dari program untuk menentukan KPK dan FPB dari dua bilangan bulat positif. Variabel dan fungsi dideklarasikan secara terpisah, tanpa ada pengelompokan yang jelas. Hal ini dapat menyebabkan kesulitan saat membaca kode dan meningkatkan risiko kesalahan program.

Kodingan kedua memiliki struktur yang lebih baik dan menggunakan konsep OOP. Dalam kodingan kedua, fungsi-fungsi dibungkus ke dalam objek-objek yang berbeda untuk meningkatkan modularitas dan memudahkan pembacaan kode. Selain itu, variabel global dihindari sebisa mungkin dan diganti dengan variabel lokal di dalam metode kelas. Ini membuat kode lebih aman dan mudah dipelihara.

Penjelasan lebih lanjut kodingan yang sudah dibenarkan:

Berikut adalah langkah kerja pada algoritma

ALGORITMA EUCLIDEAN



Gambar 3 Mencari KPK dan FPB menggunakan algoritma Euclidean. (Sumber: Penulis)

Kelas Math berisi dua buah fungsi yaitu `getKPK()` dan `getFPB()`. Fungsi `getKPK()` digunakan untuk mencari nilai KPK atau Kelipatan Persekutuan Terkecil dari dua buah bilangan. Sedangkan fungsi `getFPB()` digunakan untuk mencari nilai FPB atau Faktor Persekutuan Besar dari dua buah bilangan.

Pada bagian ini yang akan dijelaskan adalah Algoritma Euclidean yang digunakan dalam fungsi `getKPK()` dan `getFPB()`. Algoritma Euclidean adalah metode yang digunakan untuk mencari nilai FPB dari dua buah bilangan dengan menggunakan pembagian bersusun (successive division).

Algoritma Euclidean bekerja dengan cara mengulang penggantian bilangan kedua dengan sisa hasil bagi antara bilangan pertama dan bilangan kedua sampai diperoleh sisa hasil bagi nol. Sementara itu, bilangan terakhir yang bukan nol itulah yang menjadi nilai FPB dari dua bilangan awal.

Contoh penggunaan algoritma Euclidean dalam fungsi `getFPB()`:

- 1) Misalkan kita ingin mencari nilai FPB dari dua bilangan yaitu 12 dan 18. Maka, langkah-langkah yang harus dilakukan adalah sebagai berikut:
- 2) Hitung sisa hasil bagi dari pembagian 18 dengan 12, sehingga diperoleh sisa hasil bagi 6.
- 3) Ganti nilai bilangan kedua (yaitu 18) dengan sisa hasil bagi (yaitu 6) tersebut.
- 4) Hitung sisa hasil bagi dari pembagian 12 dengan 6, sehingga diperoleh sisa hasil bagi 0.
- 5) Karena sisa hasil bagi adalah nol, maka nilai FPB dari 12 dan 18 adalah bilangan terakhir yang bukan nol yaitu 6.

Dalam kode program di atas, algoritma Euclidean digunakan dalam fungsi `getFPB()` untuk mencari nilai FPB dari dua buah bilangan. Pada awalnya, `num1` merupakan bilangan pertama dan `num2` merupakan bilangan kedua. Kemudian, dilakukan perulangan `while` yang terus berjalan selama `num2` tidak sama dengan 0.

Pada setiap iterasi, nilai temporary (`temp`) diisi dengan nilai `num2`, lalu nilai `num2` diisi dengan sisa hasil bagi antara `num1` dan `num2`. Sedangkan, nilai `num1` diisi dengan nilai temporary. Proses ini dilakukan secara berulang-ulang sampai nilai `num2` menjadi 0.

Saat nilai `num2` sudah menjadi 0, nilai `num1` akan berisi nilai FPB dari dua bilangan awal. Setelah itu, nilai `num1` dikembalikan sebagai hasil dari fungsi `getFPB()`.

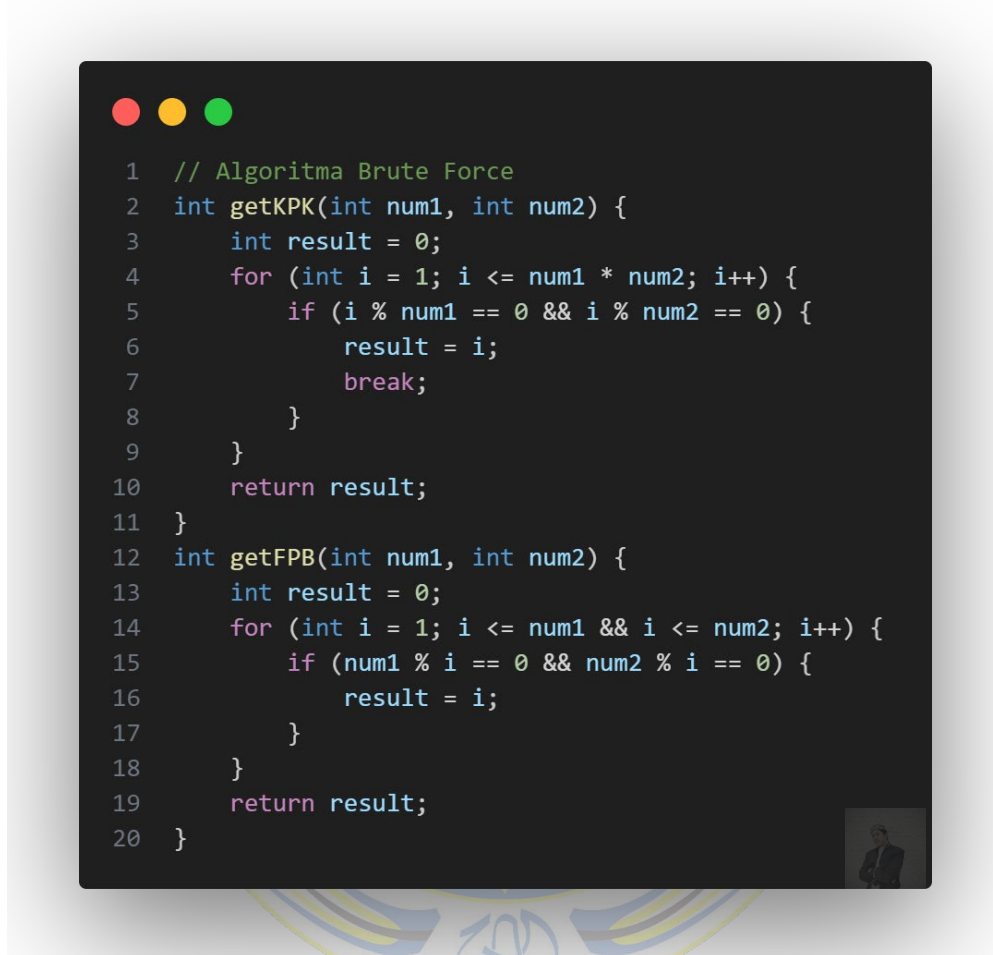
Selain itu, algoritma Euclidean juga digunakan dalam fungsi `getKPK()`. Dalam kasus ini, nilai FPB dari dua bilangan ditemukan terlebih dahulu menggunakan fungsi `getFPB()`, kemudian nilai KPK dihitung menggunakan rumus: $\text{num1} * \text{num2} / \text{FPB}$.

Sebagai contoh, jika kita ingin mencari nilai KPK dari dua bilangan yaitu 6 dan 8, maka langkah-langkah yang harus dilakukan adalah sebagai berikut:

Hitung nilai FPB dari 6 dan 8 menggunakan algoritma Euclidean, sehingga diperoleh nilai $\text{FPB} = 2$.

Hitung nilai KPK dengan cara mengalikan bilangan pertama (yaitu 6) dengan bilangan kedua (yaitu 8) dan membaginya dengan nilai FPB, sehingga diperoleh nilai $\text{KPK} = 24$.

Dalam kode program di atas, fungsi `getKPK()` menggunakan algoritma Euclidean untuk mencari nilai FPB dari dua buah bilangan `num1` dan `num2`, kemudian menghitung nilai KPK dengan rumus $\text{num1} * \text{num2} / \text{FPB}$.



Gambar 4 Mencari KPK dan FPB menggunakan algoritma Euclidean. (Sumber: Penulis)

Algoritma Brute Force adalah suatu metode penyelesaian masalah dengan cara mencoba satu per satu semua kemungkinan yang ada sampai ditemukan solusi yang benar. Pada dasarnya, algoritma Brute Force merupakan metode pencarian pola secara sistematis dalam sebuah ruang pencarian. Algoritma ini cocok digunakan untuk menyelesaikan masalah yang memiliki batasan ruang pencarian relatif kecil.

Salah satu contoh penerapan algoritma Brute Force adalah pada fungsi `getKPK` dan `getFPB` dalam program di atas. Fungsi `getKPK` bertujuan untuk mencari Kelipatan

Persekutuan Kecil (KPK) dari dua bilangan, sedangkan fungsi getFPB bertujuan untuk mencari Faktor Persekutuan Besar (FPB) dari dua bilangan.

Berikut adalah alur kerja dari algoritma Brute Force pada fungsi getKPK(num1, num2):

- 1) Tentukan dua bilangan num1 dan num2.
- 2) Inisialisasi variabel result dengan nilai 0.
- 3) Lakukan iterasi pada semua bilangan dari 1 hingga $\text{num1} * \text{num2}$:
 - a. Cek apakah bilangan tersebut habis dibagi oleh num1 dan num2.
 - b. Jika bilangan tersebut habis dibagi oleh num1 dan num2, maka simpan bilangan tersebut ke dalam variabel result dan keluar dari loop.
- 4) Return hasil variabel result sebagai KPK dari dua bilangan tersebut.

Sebagai contoh, jika ingin menentukan KPK dari bilangan 6 dan 8 menggunakan algoritma Brute Force, maka akan dicoba semua bilangan dari 1 sampai 48 ($6 * 8$) secara berurutan, dan bilangan yang pertama kali ditemukan yang habis dibagi oleh 6 dan 8 akan menjadi hasilnya, yaitu 24.

Berikut adalah alur kerja dari algoritma Brute Force pada fungsi getFPB(num1, num2):

- 1) Tentukan dua bilangan num1 dan num2.
- 2) Inisialisasi variabel result dengan nilai 0.
- 3) Lakukan iterasi pada semua bilangan dari 1 hingga bilangan terkecil antara num1 dan num2: a. Cek apakah bilangan tersebut dapat membagi habis num1 dan num2. b. Jika bilangan tersebut dapat membagi habis num1 dan num2, maka simpan bilangan tersebut ke dalam variabel result.
- 4) Return hasil variabel result sebagai FPB dari dua bilangan tersebut.

Sebagai contoh, jika ingin menentukan FPB dari bilangan 18 dan 24 menggunakan algoritma Brute Force, maka akan dicoba semua bilangan dari 1 sampai 18 (karena itu bilangan terkecil antara 18 dan 24) secara berurutan, dan bilangan yang pertama kali ditemukan yang dapat membagi habis kedua bilangan tersebut akan menjadi hasilnya, yaitu 6.

Namun, meskipun algoritma Brute Force cukup mudah dipahami dan mudah diimplementasikan, terdapat beberapa kekurangan yang membuat algoritma ini kurang efisien, terutama untuk masalah dengan skala yang besar. Kelemahan utama dari algoritma Brute Force adalah kompleksitas waktu yang tinggi karena mencoba semua kemungkinan solusi secara sistematis. Oleh karena itu, algoritma Brute Force kurang cocok digunakan untuk menyelesaikan masalah dengan skala yang besar.

Algoritma Euclidean, pada saat yang sama, adalah algoritma yang digunakan untuk mencari FPB dan KPK dari dua bilangan dengan lebih efisien dibandingkan algoritma Brute Force. Algoritma ini menggunakan konsep bahwa FPB atau KPK dari dua bilangan dapat dicari dengan mengulang pemakaian operasi modulo (sisa bagi) pada kedua bilangan tersebut.

Berikut adalah alur kerja dari algoritma Euclidean pada fungsi getKPK(num1, num2):

- 1) Tentukan dua bilangan num1 dan num2.
- 2) Cari nilai FPB dari dua bilangan tersebut menggunakan algoritma Euclidean (lihat fungsi getFPB).
- 3) Hitung hasil KPK dari dua bilangan dengan rumus: $(\text{num1} * \text{num2}) / \text{fpb}$.
- 4) Return hasil KPK sebagai output.

Berikut adalah alur kerja dari algoritma Euclidean pada fungsi getFPB (num1, num2):

- 1) Tentukan dua bilangan num1 dan num2.
- 2) Lakukan operasi modulo pada kedua bilangan untuk mendapatkan sisa hasil bagi.
- 3) Jika sisa hasil bagi adalah 0, maka FPB adalah bilangan yang lebih kecil dari kedua bilangan tersebut.
- 4) Jika sisa hasil bagi bukan 0, maka FPB kembali dicari dengan menggunakan algoritma Euclidean pada bilangan yang lebih kecil dan sisa hasil bagi sebelumnya.
- 5) Ulangi langkah 2-4 sampai ditemukan nilai FPB.
- 6) Return nilai hasil FPB sebagai output.

Sebagai contoh, jika ingin menentukan FPB dari bilangan 18 dan 24 menggunakan algoritma Euclidean, maka dapat dilakukan sebagai berikut:

- Lakukan operasi modulo 18 dan 24. Hasilnya adalah 18.
- Lakukan operasi modulo 24 dan 18. Hasilnya adalah 6.
- Lakukan operasi modulo 18 dan 6. Hasilnya adalah 0.

Karena sisa hasil bagi adalah 0, maka FPB adalah bilangan yang lebih kecil dari kedua bilangan tersebut, yaitu 6.

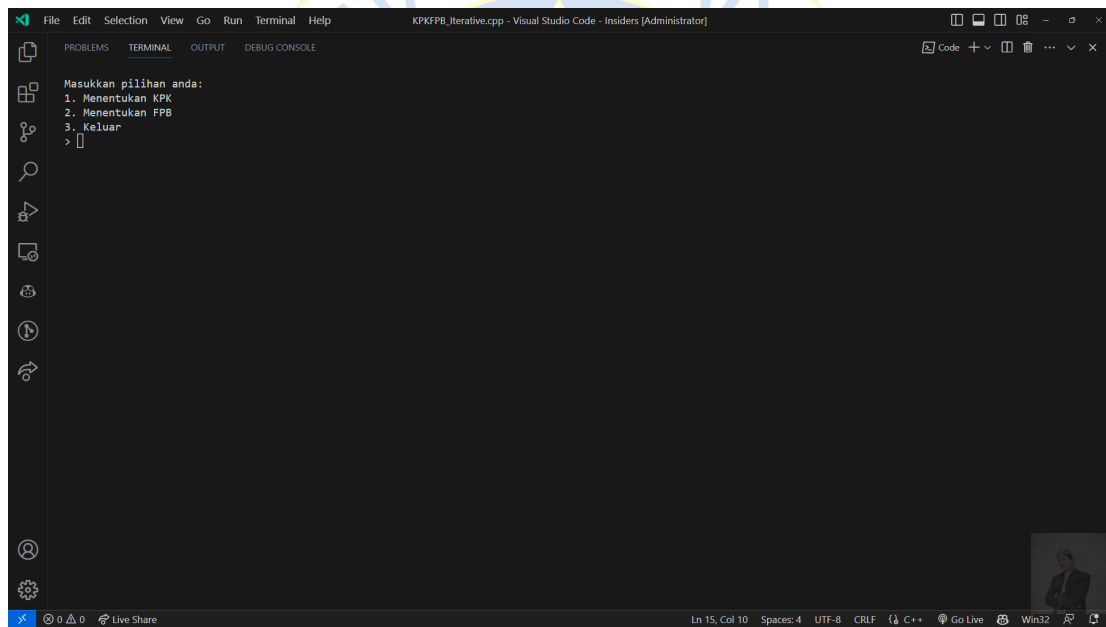
Dapat disimpulkan bahwa algoritma Euclidean lebih efektif dibandingkan algoritma Brute Force dalam mencari FPB dan KPK dari dua bilangan karena kompleksitas waktu yang lebih rendah. Algoritma Euclidean hanya membutuhkan beberapa iterasi saja untuk mendapatkan hasil KPK atau FPB, sedangkan algoritma Brute Force membutuhkan iterasi sebanyak kemungkinan bilangan yang harus dicoba. Selain itu, algoritma Euclidean juga dapat bekerja pada masalah dengan skala yang besar, sedangkan algoritma Brute Force kurang cocok untuk masalah dengan skala yang besar karena kompleksitas waktu yang tinggi. Oleh karena itu, algoritma Euclidean lebih sering digunakan dalam aplikasi nyata untuk menyelesaikan masalah FPB dan KPK.

Fungsi getKPK(num1, num2) pada program di atas menggunakan algoritma Brute Force untuk mencari KPK dari dua bilangan. Pertama-tama, fungsi ini menginisialisasi variabel result dengan nilai 0. Kemudian, dilakukan iterasi pada semua bilangan dari 1 hingga $\text{num1} * \text{num2}$. Pada setiap iterasi, fungsi akan memeriksa apakah bilangan tersebut habis dibagi oleh num1 dan num2. Jika ya, maka bilangan tersebut akan disimpan ke dalam variabel result dan loop akan berakhir. Fungsi kemudian mengembalikan nilai variabel result sebagai hasil KPK.

Fungsi `getFPB(num1, num2)` pada program di atas juga menggunakan algoritma Brute Force untuk mencari FPB dari dua bilangan. Pertama-tama, fungsi ini menginisialisasi variabel `result` dengan nilai 0. Kemudian, dilakukan iterasi pada semua bilangan dari 1 hingga bilangan terkecil antara `num1` dan `num2`. Pada setiap iterasi, fungsi akan memeriksa apakah bilangan tersebut dapat membagi habis `num1` dan `num2`. Jika ya, maka bilangan tersebut akan disimpan ke dalam variabel `result`. Setelah seluruh iterasi selesai, fungsi akan mengembalikan nilai variabel `result` sebagai hasil FPB.

Dalam kodingan di atas, fungsi `getKPK` dan `getFPB` yang menggunakan algoritma Euclidean di-comment karena tidak digunakan. Namun, jika ingin mencoba menggunakan algoritma Euclidean, cukup un-comment kedua fungsi tersebut dan comment fungsi-fungsi Brute Force.

Berikut adalah hasil output kedua kodingannya:



Gambar 5 Tampilan menu. (Sumber: Penulis)

Berikut adalah contoh output KPK.

The image displays two screenshots of a Visual Studio Code terminal window. The terminal window has a dark theme and shows the output of a C++ program. The program prompts the user to enter two numbers, calculates their Least Common Multiple (KPK), and displays the result.

First Screenshot:

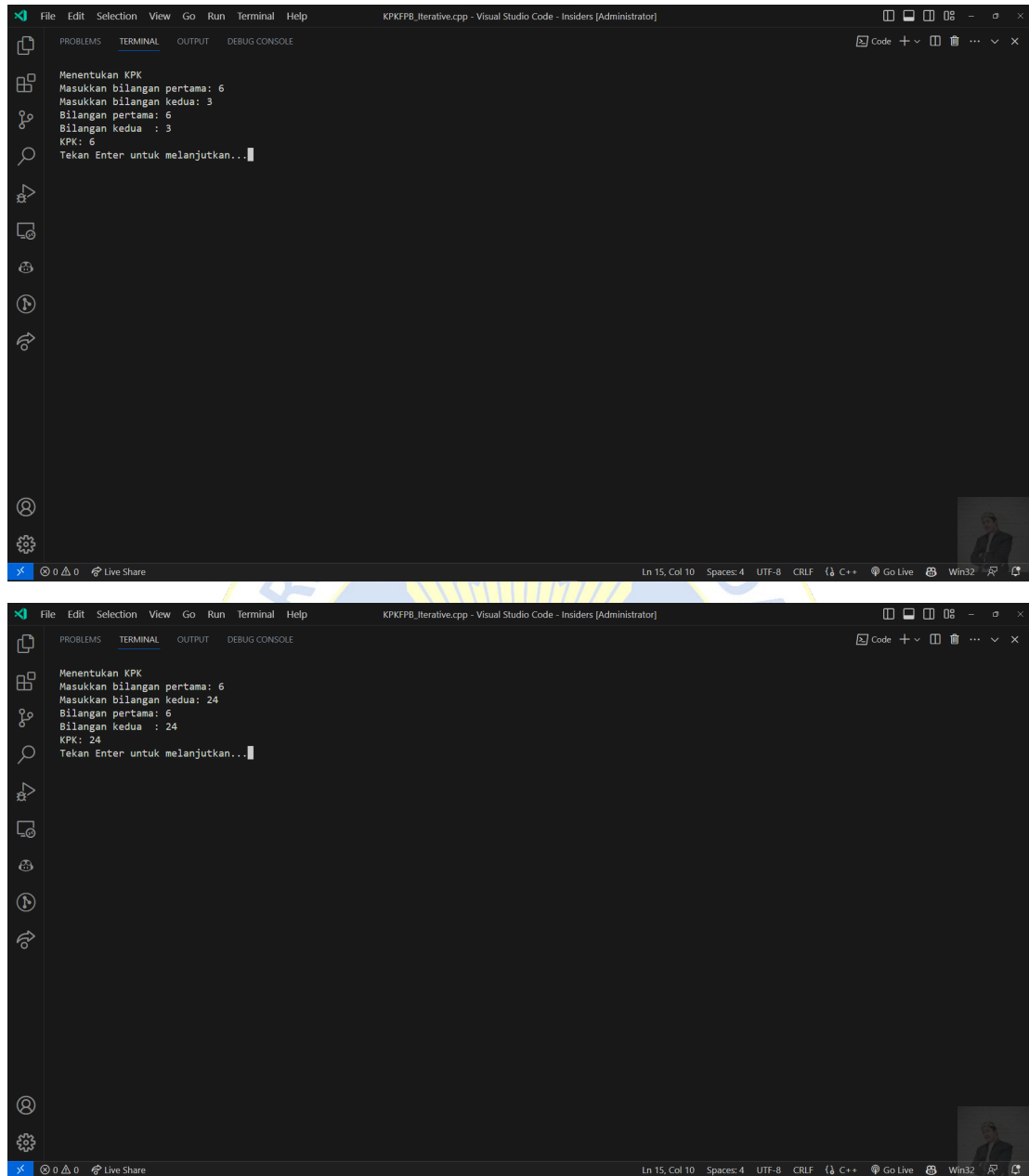
```

Menentukan KPK
Masukkan bilangan pertama: 25
Masukkan bilangan kedua: 27
Bilangan pertama: 25
Bilangan kedua : 27
KPK: 675
Tekan Enter untuk melanjutkan...
  
```

Second Screenshot:

```

Menentukan KPK
Masukkan bilangan pertama: 4
Masukkan bilangan kedua: 7
Bilangan pertama: 4
Bilangan kedua : 7
KPK: 28
Tekan Enter untuk melanjutkan...
  
```



The image displays two screenshots of a Visual Studio Code terminal window. The terminal window is titled "KPKFPB_iterative.cpp - Visual Studio Code - Insiders [Administrator]". The terminal output shows the execution of a C++ program that calculates the Least Common Multiple (KPK) of two input numbers.

Top Screenshot:

```

Menentukan KPK
Masukkan bilangan pertama: 6
Masukkan bilangan kedua: 3
Bilangan pertama: 6
Bilangan kedua : 3
KPK: 6
Tekan Enter untuk melanjutkan...

```

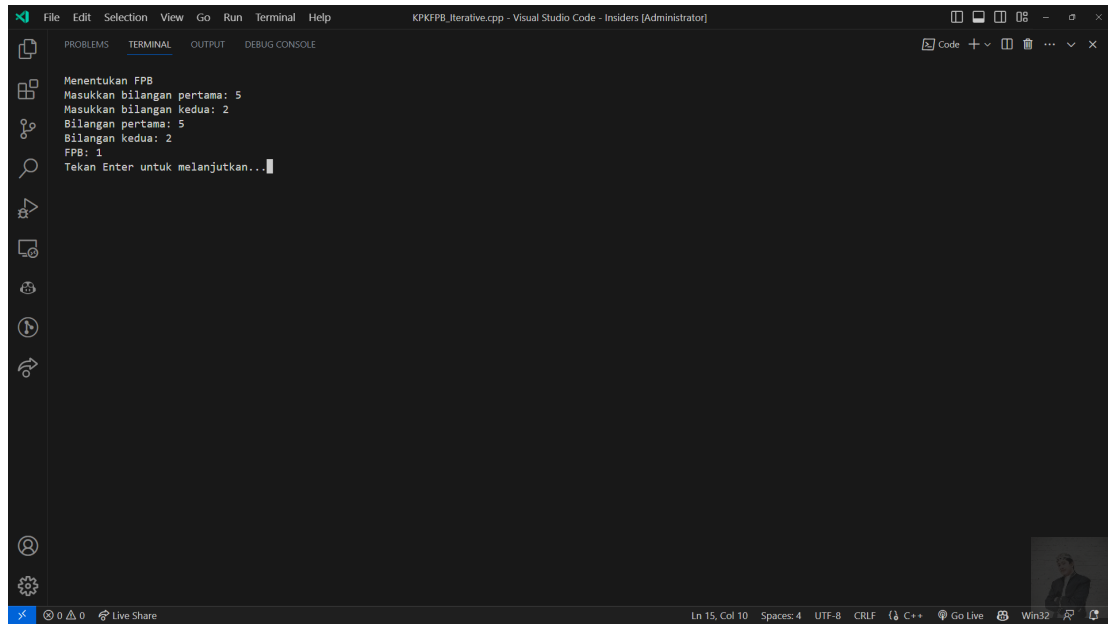
Bottom Screenshot:

```

Menentukan KPK
Masukkan bilangan pertama: 6
Masukkan bilangan kedua: 24
Bilangan pertama: 6
Bilangan kedua : 24
KPK: 24
Tekan Enter untuk melanjutkan...

```

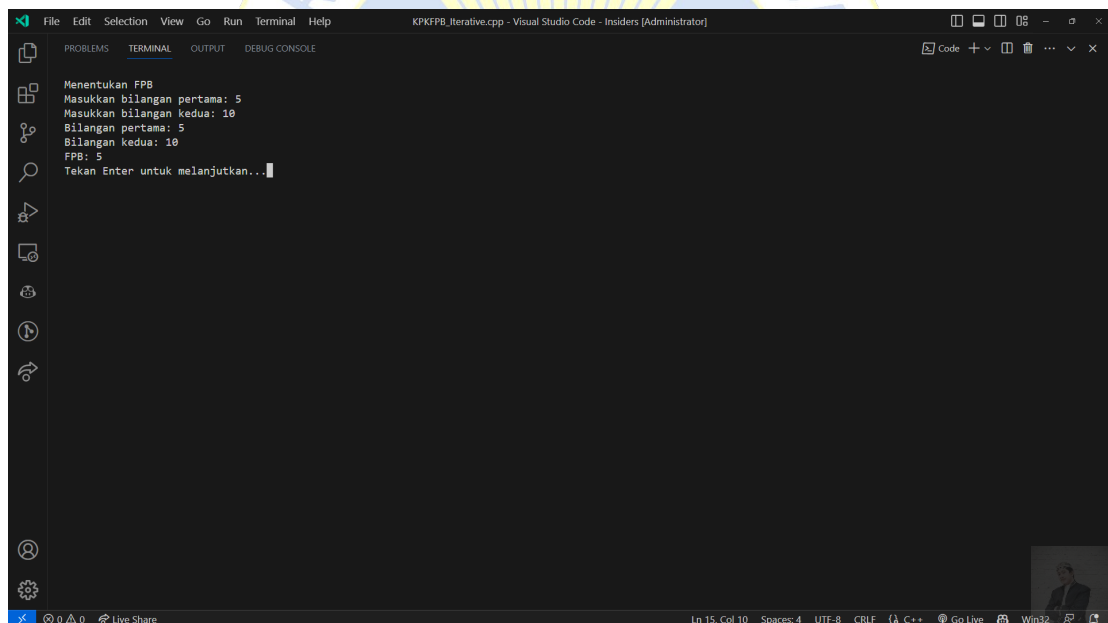
Gambar 6 Output KPK. (Sumber: Penulis)



```

KPKFPB_iterative.cpp - Visual Studio Code - Insiders [Administrator]
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE
Menentukan FPB
Masukkan bilangan pertama: 5
Masukkan bilangan kedua: 2
Bilangan pertama: 5
Bilangan kedua: 2
FPB: 1
Tekan Enter untuk melanjutkan...

```



```

KPKFPB_iterative.cpp - Visual Studio Code - Insiders [Administrator]
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE
Menentukan FPB
Masukkan bilangan pertama: 5
Masukkan bilangan kedua: 10
Bilangan pertama: 5
Bilangan kedua: 10
FPB: 5
Tekan Enter untuk melanjutkan...

```

The image consists of two screenshots of a Visual Studio Code terminal window. The terminal window is titled 'KPKFPB_iterative.cpp - Visual Studio Code - Insiders [Administrator]'. The terminal output is as follows:

```

Menentukan FPB
Masukkan bilangan pertama: 24
Masukkan bilangan kedua: 6
Bilangan pertama: 24
Bilangan kedua: 6
FPB: 6
Tekan Enter untuk melanjutkan...

```

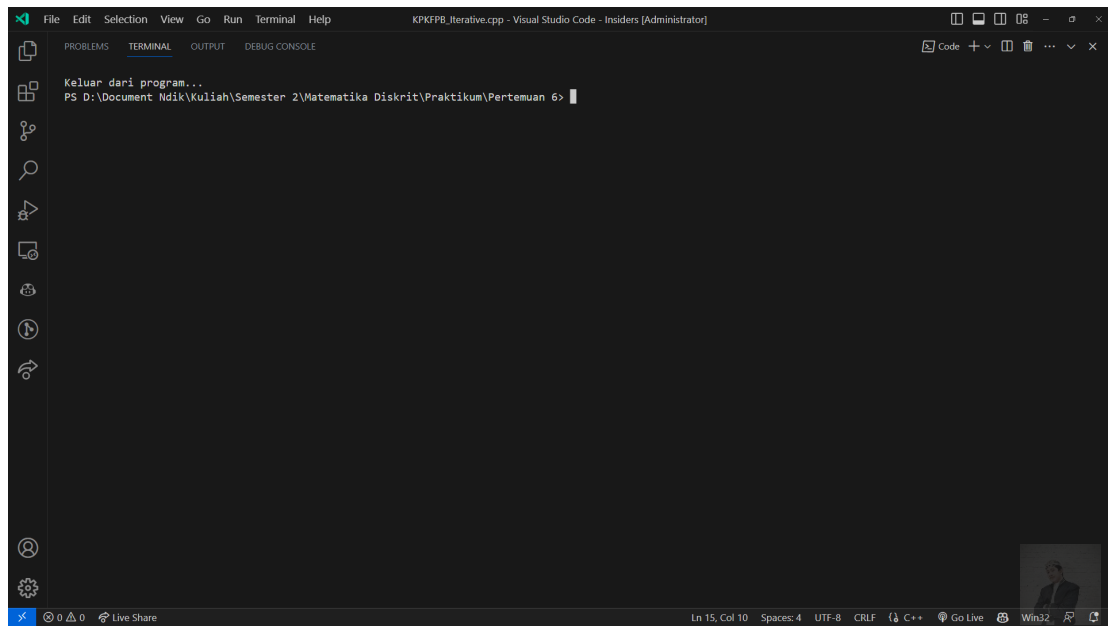
The second screenshot shows the same terminal window with the following output:

```

Menentukan FPB
Masukkan bilangan pertama: 6
Masukkan bilangan kedua: 240
Bilangan pertama: 6
Bilangan kedua: 240
FPB: 6
Tekan Enter untuk melanjutkan...

```

Gambar 7 Output FPB. (Sumber: Penulis)



```

KPKFPB_iterative.cpp - Visual Studio Code - Insiders [Administrator]
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE

Keluar dari program...
PS D:\Document Ndik\Kuliah\Semester 2\Matematika Diskrit\Praktikum\Pertemuan 6>
    
```

Gambar 8 Saat memencet 3 Keluar. (Sumber: Penulis)

Untuk mengakses kodingan, dapat mengakses link github berikut.

<https://github.com/IRedDragonICY/Matematika-Diskrit>

