**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:



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

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



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

Kodingan pertama yang terdapat pada modul memiliki beberapa kesalahan yang mungkin diakibatkan oleh ketidaktelitian 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.

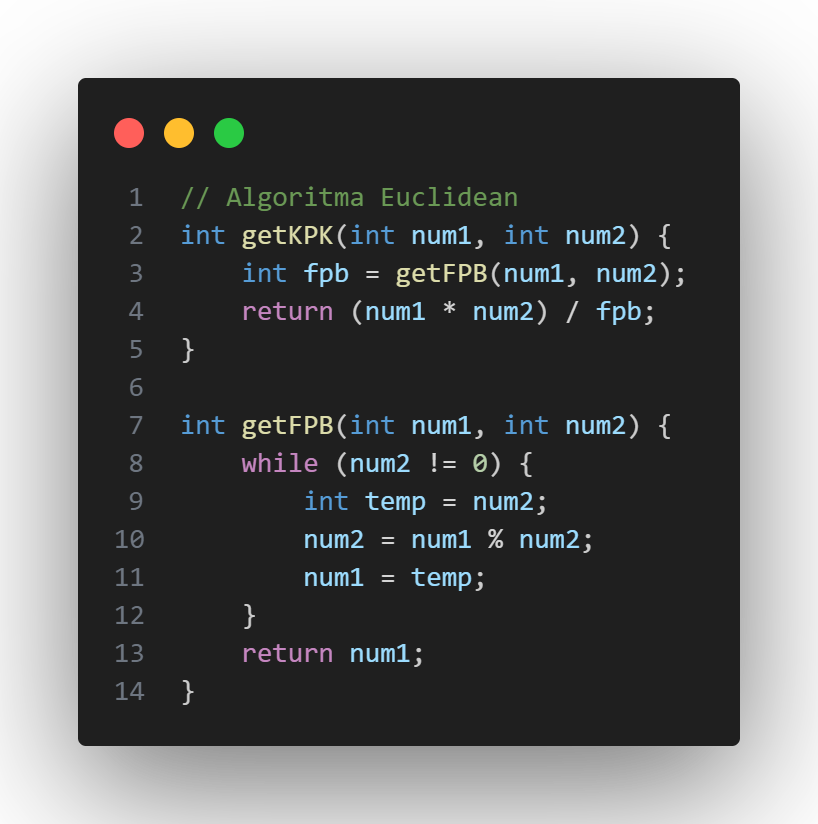
Kelas Menu berfungsi untuk menampilkan menu pilihan pada layar dan meminta user untuk memasukkan dua bilangan yang akan diproses. Kemudian, Program class menjalankan loop while yang tak henti-hentinya sampai user memilih opsi "Keluar" dari menu. Program Class kemudian memanggil fungsi yang sesuai dari 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().

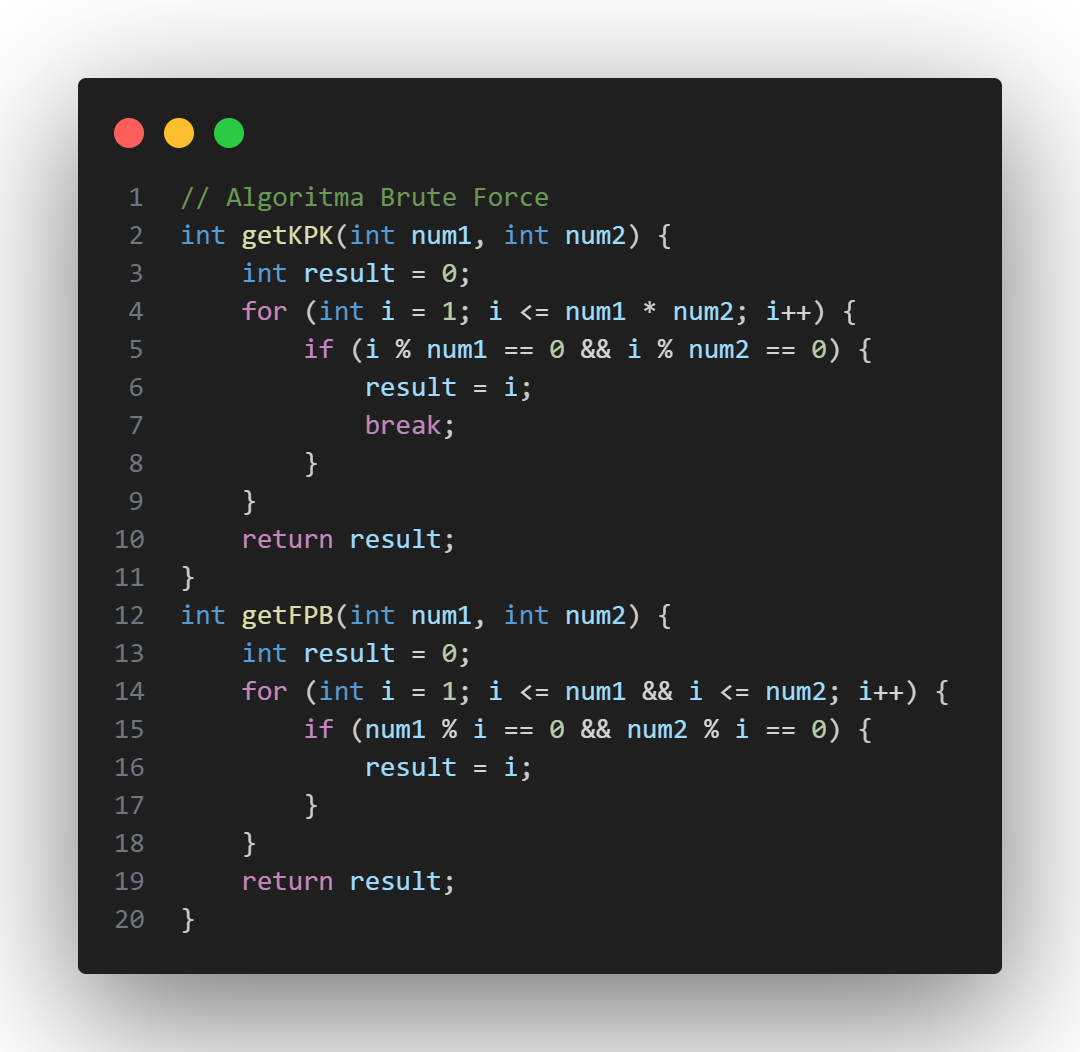
Selain itu, algoritma Euclidean juga digunakan dalam fungsi getKPK(). Dalam kasus ini, nilai FPB dari dua bilangan ditemukan terlebih dahulu menggunakan fungsi getFPB(), kemudian nilai KPK dihitung menggunakan rumus: num1 \* num2 / FPB.

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 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 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 num1 \* num2 / 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 num1\*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.

1. 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: (num1 \* num2) / 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.

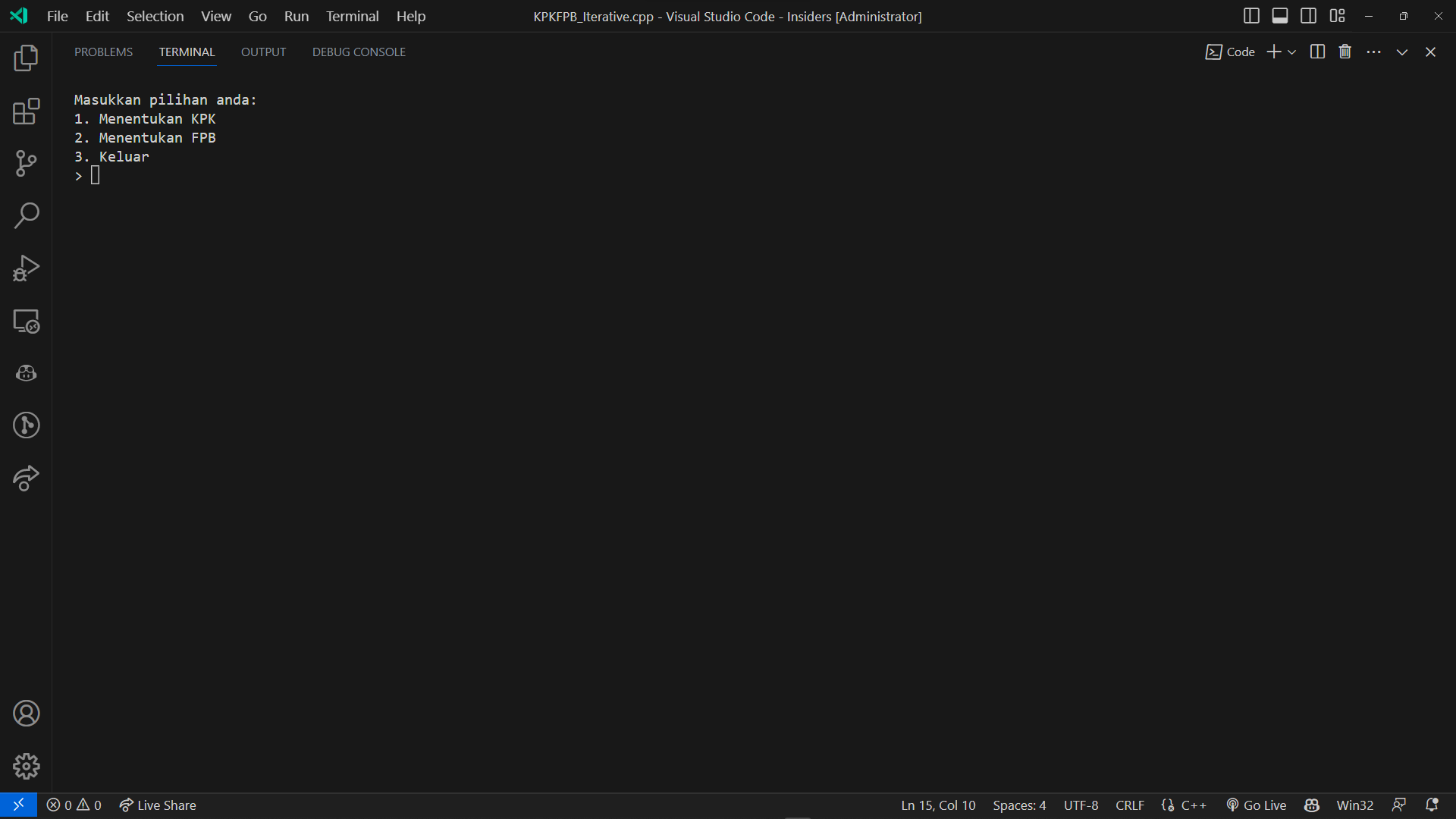
Dapat disimpulkan bahwa algoritma Euclidean lebih efektif dibandingkan algoritma Brute Force dalam mencari FPB dan KPK dari dua bilangan karena kompleksitas waktu yang lebih rendah. Algoritma Euclidean hanya membutuhkan beberapa iterasi saja untuk mendapatkan hasil KPK atau FPB, sedangkan algoritma Brute 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 num1\*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.

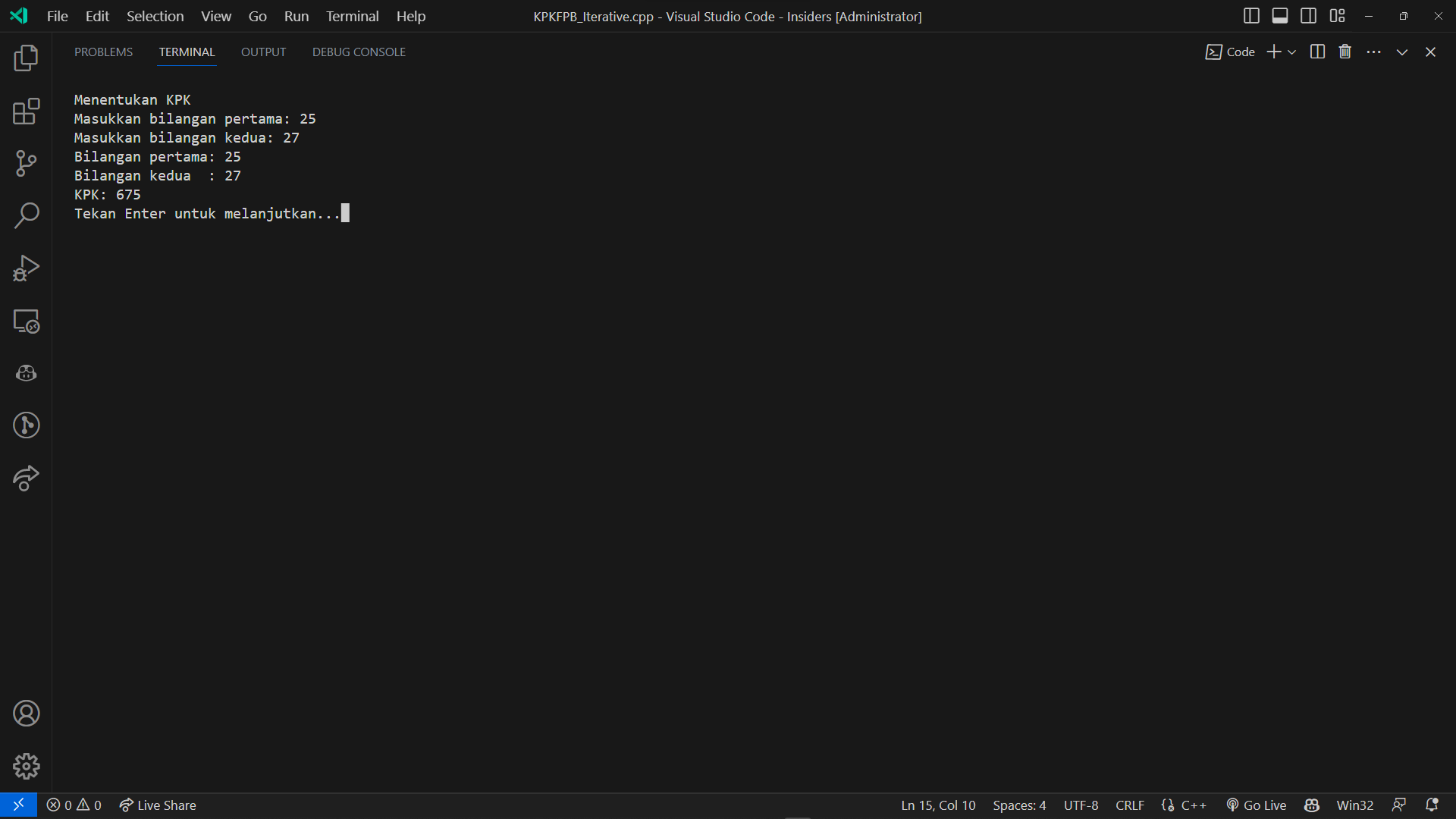
Dalam kodingan di atas, fungsi getKPK dan getFPB yang menggunakan algoritma Euclidean di-comment karena tidak digunakan. Namun, jika ingin mencoba menggunakan algoritma Euclidean, cukup un-comment kedua fungsi tersebut dan comment fungsi-fungsi Brute Force.

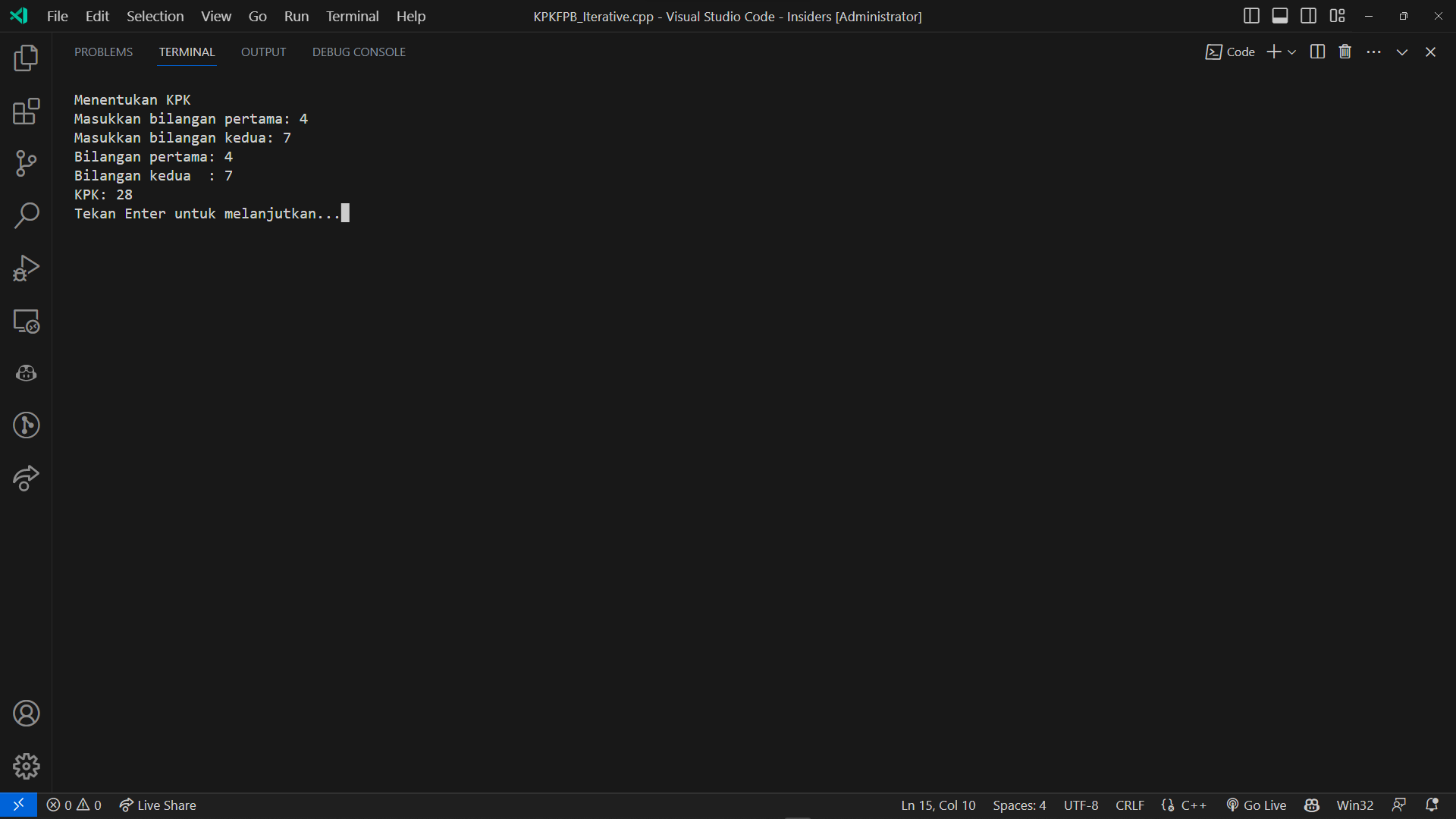
Berikut adalah hasil output kedua kodingannya:

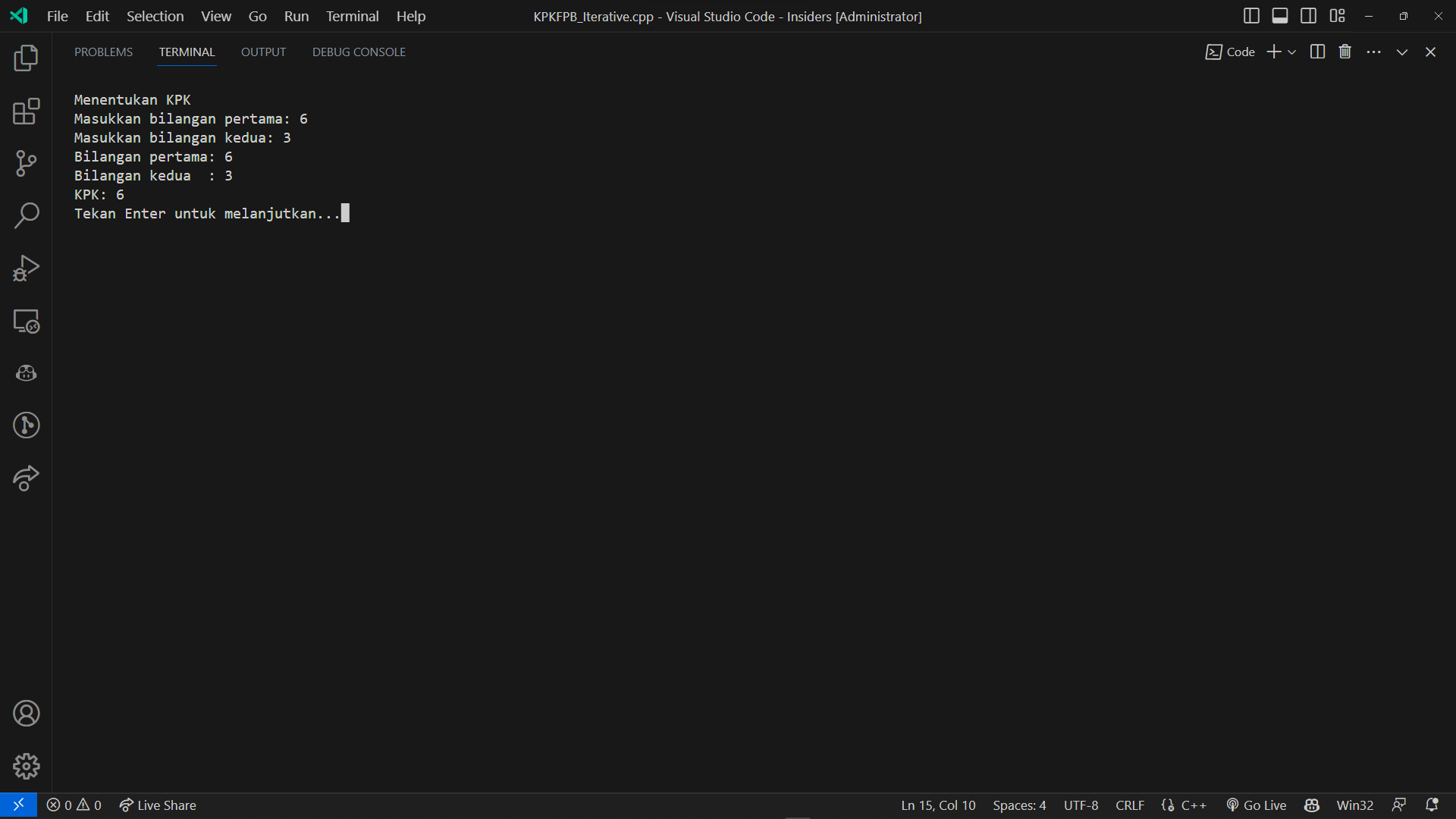


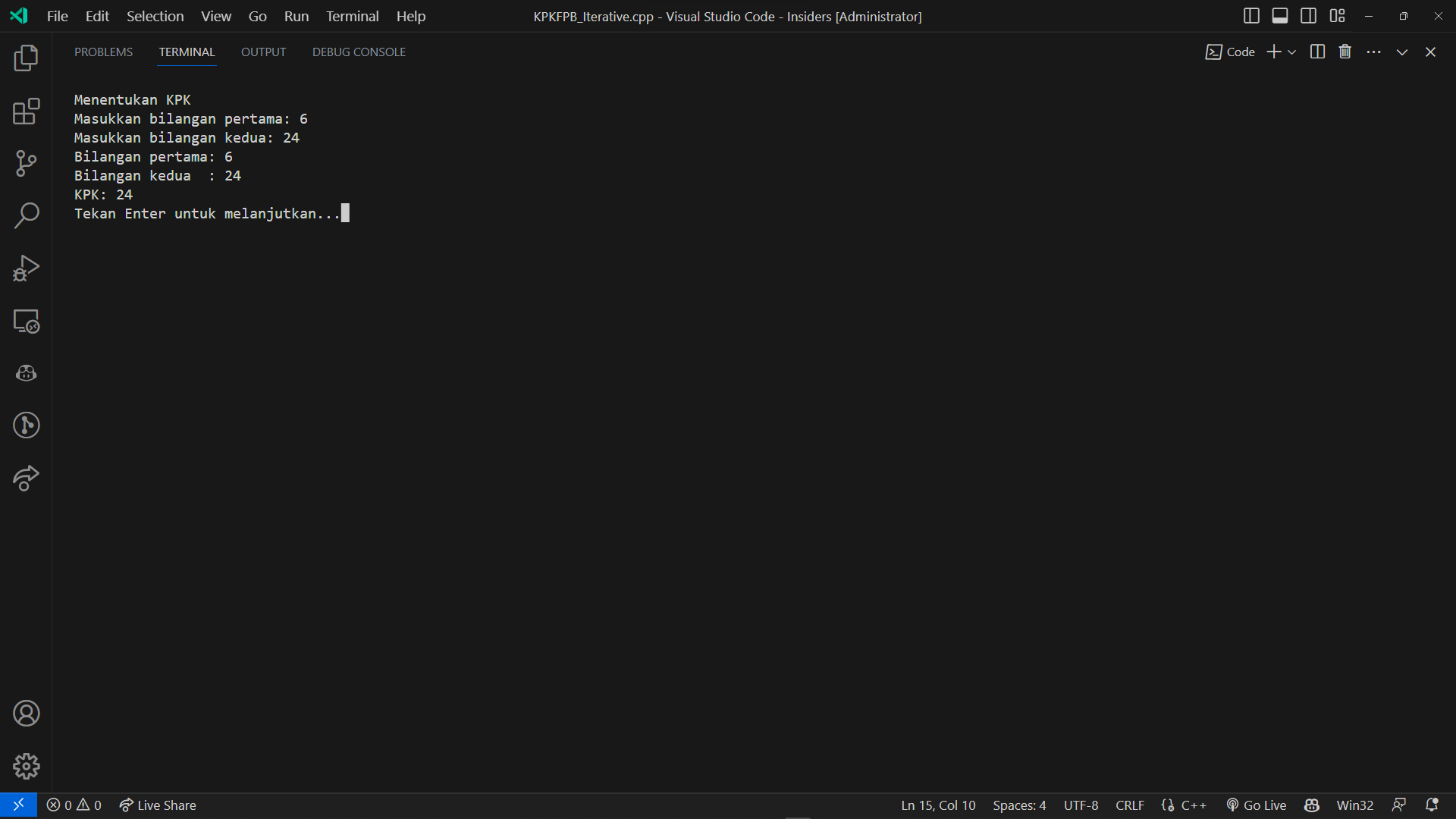
Gambar 5 Tampilan menu. (Sumber: Penulis)

Berikut adalah contoh output KPK.

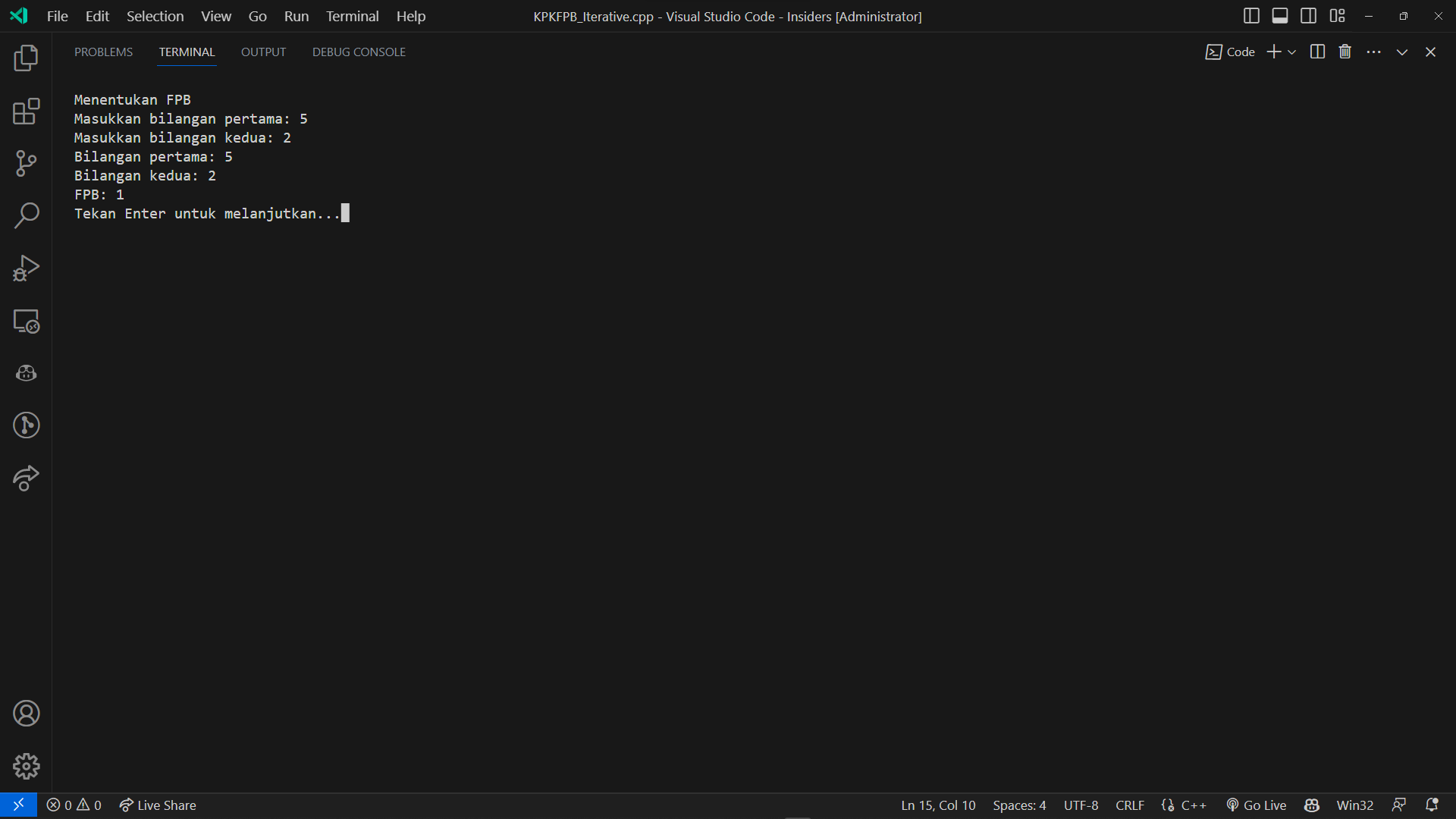


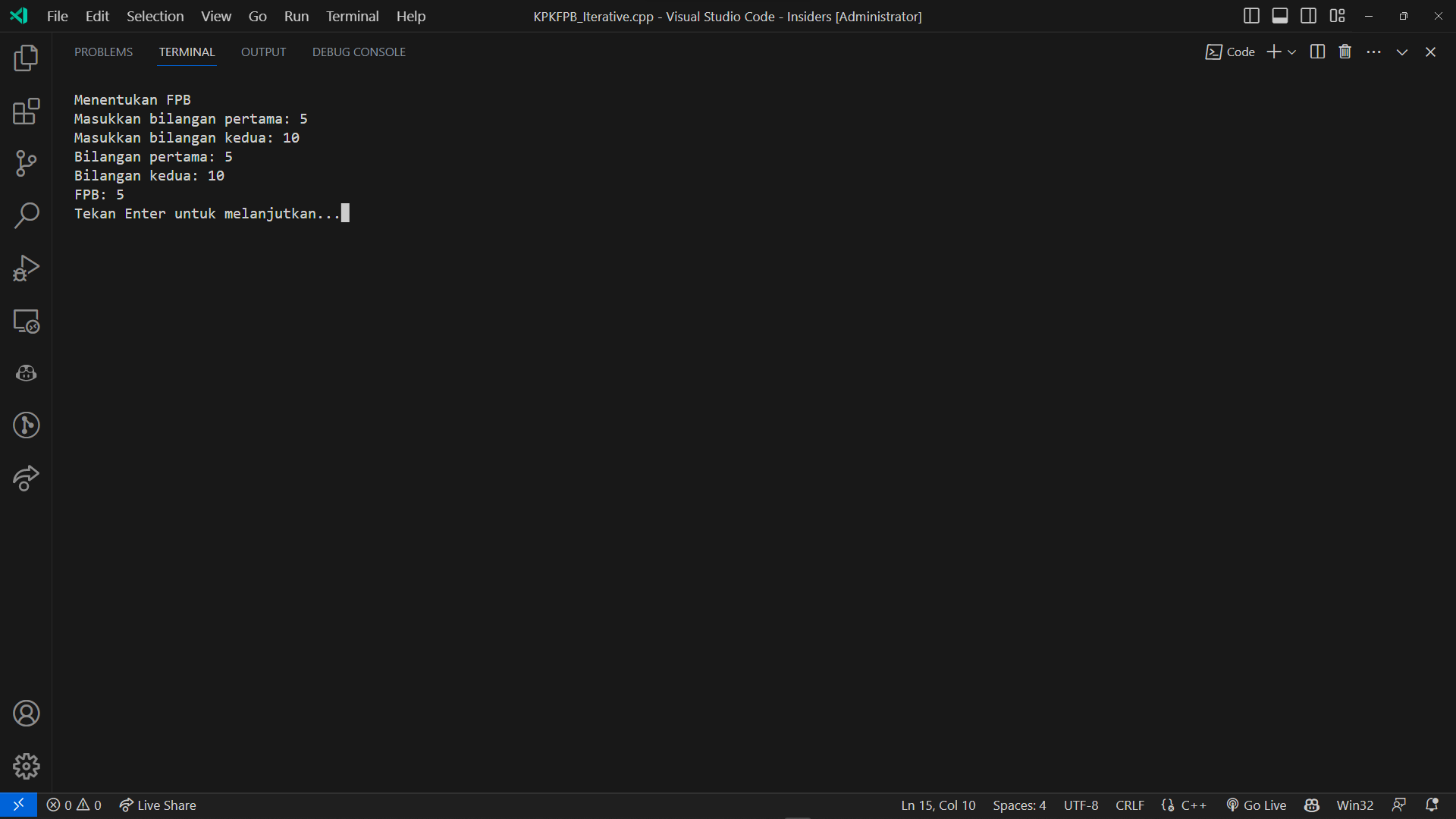


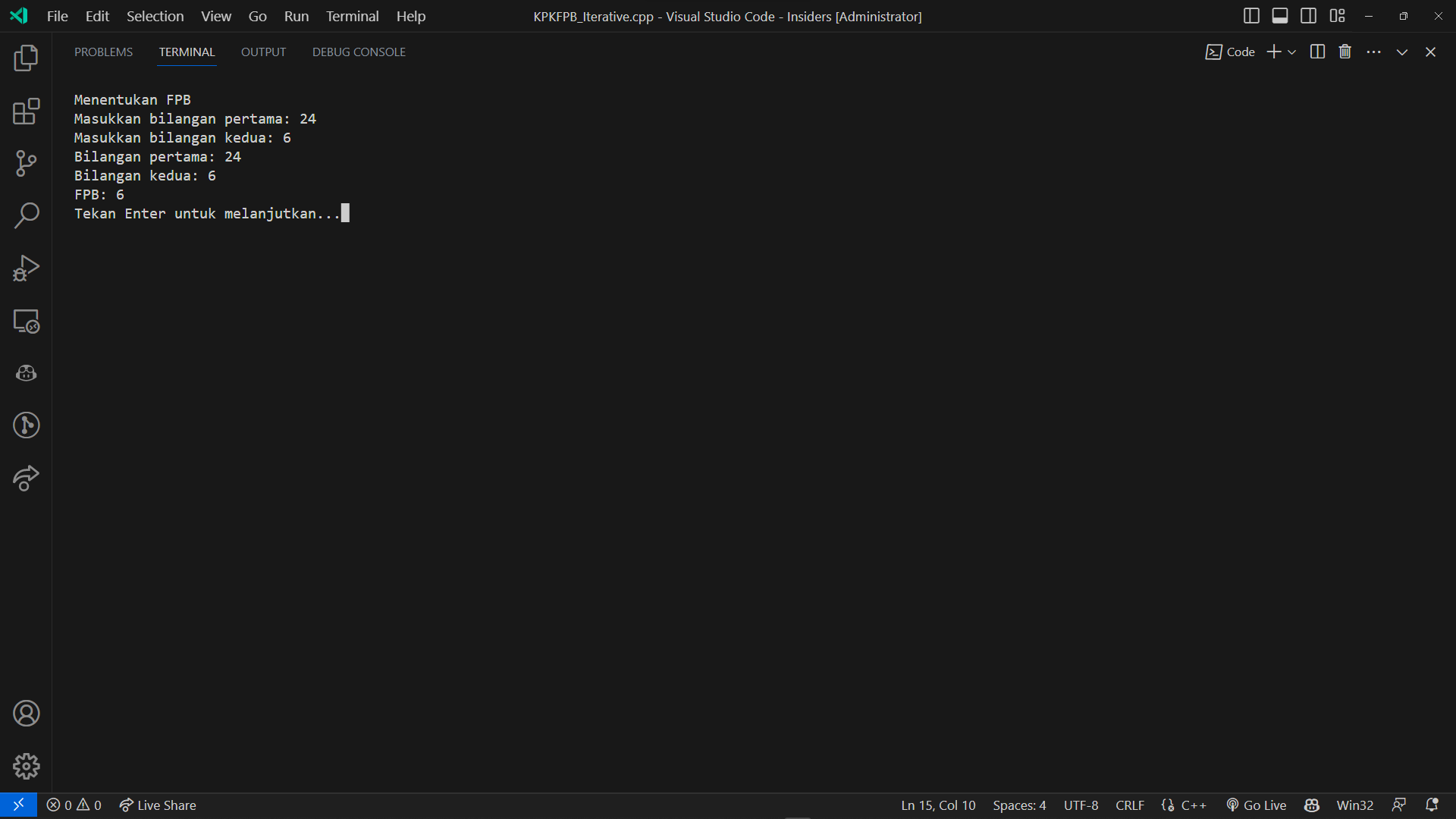


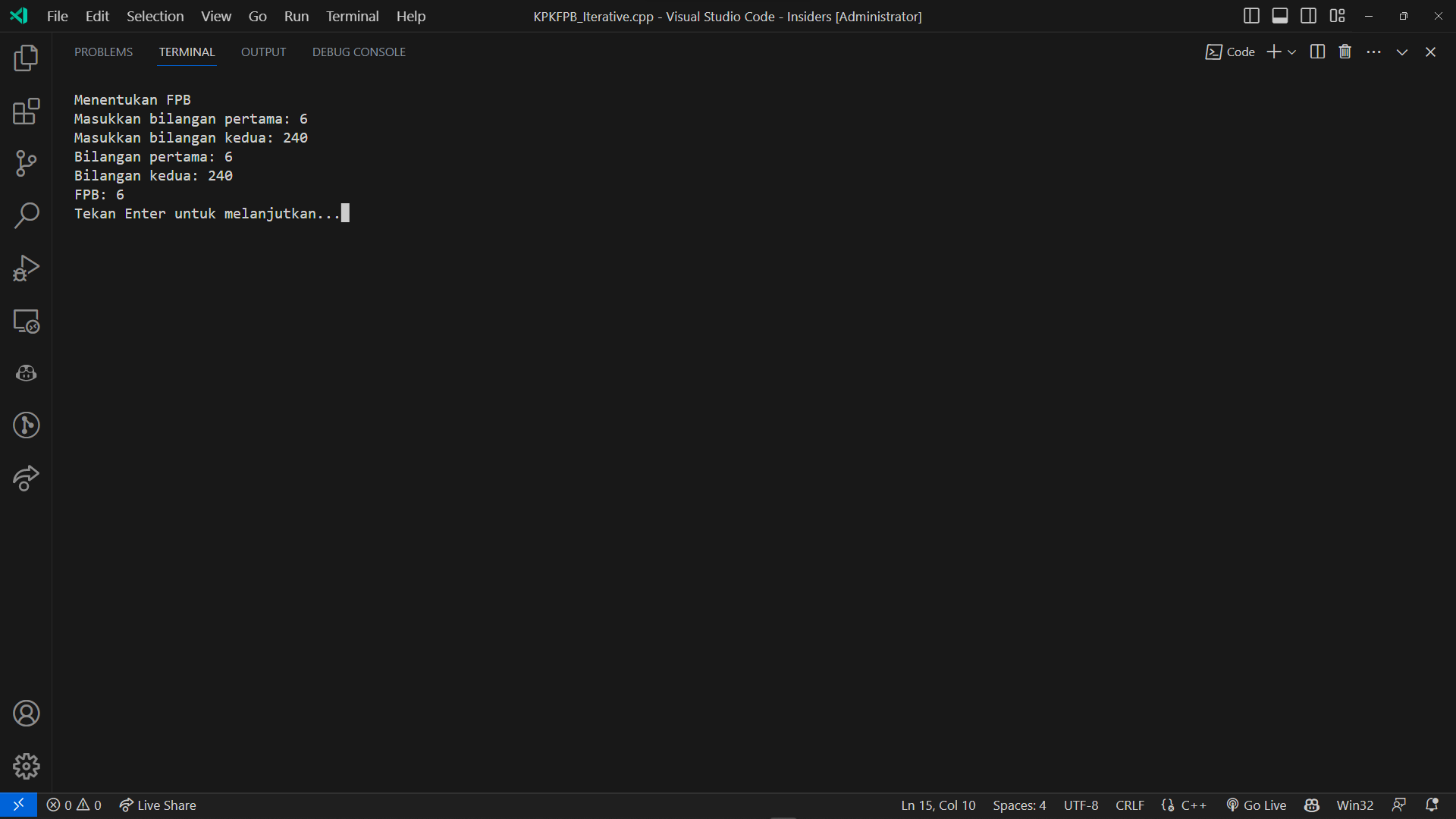


Gambar 6 Output KPK. (Sumber: Penulis)

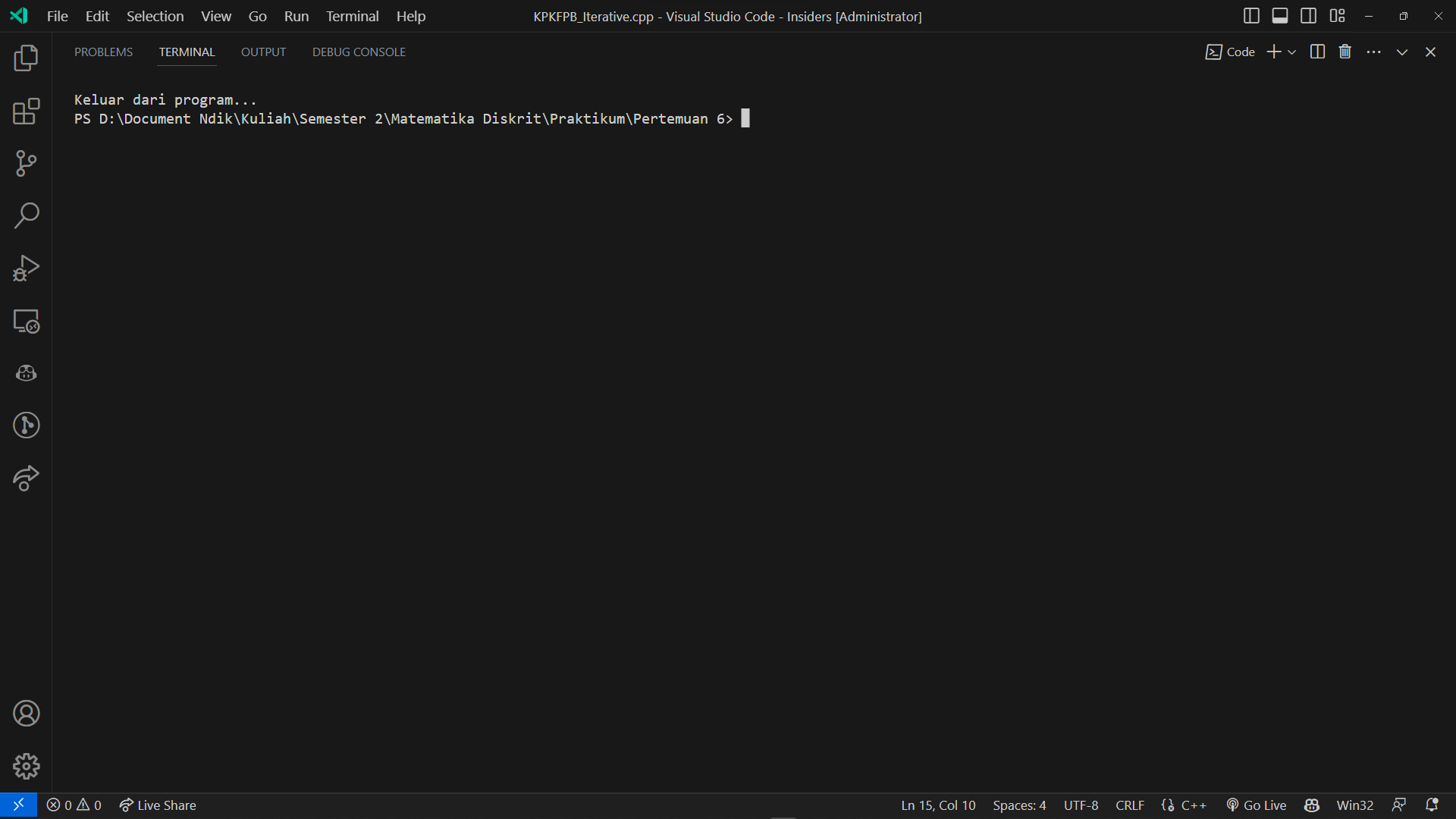








Gambar 7 Output FPB. (Sumber: Penulis)



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

Untuk mengakses kodingan, dapat mengakses link github berikut.

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