**RESUME 1**

**“PEMOGRAMAN BERGERAK”**

****

**Disusun Oleh:**

**Nama :Delia Julianti**

**Nim :21076042**

**Dosen :** **Yasdinul Huda, S.Pd.,M.T.**

**PROGRAM STUDI PENDIDIKAN TEKNIK INFORMATIKA**

**JURUSAN TEKNIK ELEKTRONIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS NEGERI PADANG**

**2023**

**“KOTLIN DASAR”**

1. **Sejarah Kotlin**

Kotlin merupakan Bahasa pemograman yang dikembangkan oleh Perusahaan Jet Brain sejak tahun 2011. Tujuan utama dalam penciptaan Kotlin adalah mengatasi beberapa kelemahan bahasa pemrograman Java sambil tetap mempertahankan kompatibilitas dengan kode Java yang sudah ada. Kotlin resmi dirilis pada tahun 2016 dengan versi 1.0. Pada tahun 2017, Google mengumumkan dukungan resmi terhadap Kotlin sebagai bahasa pemrograman yang disetujui untuk pengembangan aplikasi Android, mengukuhkan posisinya di dunia pengembangan perangkat lunak. Sejak saat itu kotlin menjadi semakin populer dan diminati oleh developer, khususnya android developer.

1. **Dasar Kotlin**

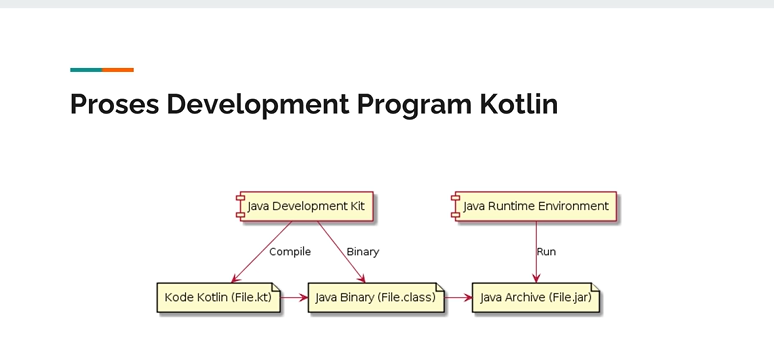
Kotlin merupakan Bahasa pemograman modern yang bersifat open source, kotlin salah satu bahasa pemograman yang digunakan dalam pembuatan aplikasi android yang dapat berjalan di JVM. Bahasa kotlin ini dikembangkan dengan sebaik mungkin agar penggunaannya dapat berdampingan dengan java.

Alasan mengapa kita menggunakan Bahasa kotlin atau mempelajarinya yaitu:

* Dapat mengatasi null: yang mana menjadi pembeda antara kotlin dan java karna jika dijava terdapat null maka akan terjadi error tetapi dikotlin kita dapat mencegahnya dengan menambahkan operator pada variable Yng mungkin null
* Lebih ringkas: Kotlin lebih ringkas karena kita dapat menghemat waktu. Hal tersebut karena tidak perlu menghabiskan waktu untuk menulis boilerplate, seperti pada Java.
* Tidak ada overhead runtime: pada Kotlin tidak ada overhead runtime hal tersebut membuat performa Kotlin tetap stabil walaupun kotlin memiliki fitur lebih banyak dari java
* Kotlin sudah didukung oleh google dan memiliki ekosistem teknologi pendukung java sudah sangat besar dan dewasa
* Fleksibel atau multiplatform
* Mudah dipelajari

**SDK dang IDE dalam kotlin:**

SDK (Software Development Kit) Kotlin adalah paket perangkat lunak yang menyediakan alat, pustaka, dan kompilator yang diperlukan untuk mengembangkan aplikasi dengan bahasa pemrograman Kotlin. SDK yang digunakan yaitu Java Development Kit versi 8 keatas.



IDE (Integrated Development Environment) Kotlin adalah perangkat lunak pengembangan yang dioptimalkan untuk bahasa Kotlin, yang terkenal dengan JetBrains IntelliJ IDEA. IDE Kotlin menyediakan berbagai fitur cerdas seperti penyelesaian otomatis kode, pengecekan kesalahan real-time, dan alat refaktor otomatis untuk meningkatkan produktivitas pengembangan.

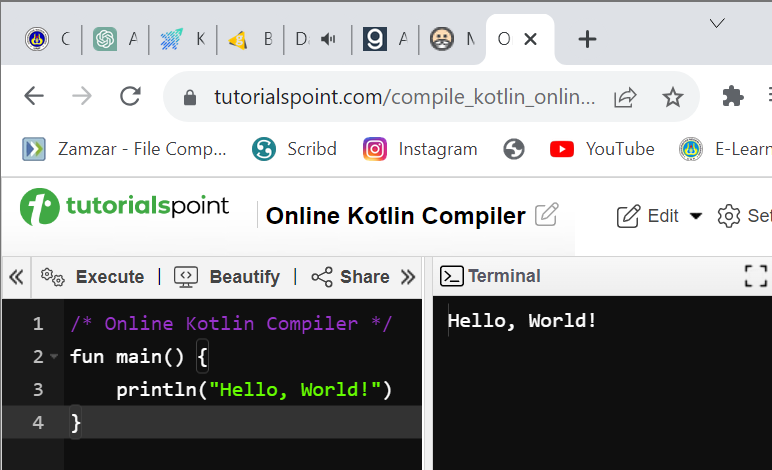
Kode: Hello World

/\* Online Kotlin Compiler \*/

fun main() {

println("Hello, World!")

}



1. **Tipe Data Number**

* **Integer number**

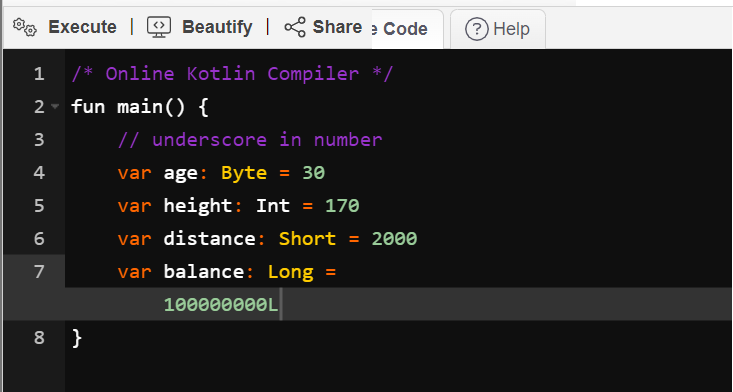
Terdapat dua tipe data dasar yang digunakan untuk menyimpan bilangan bulat, yaitu `Int` dan `Long`.

Int adalah tipe data yang digunakan untuk menyimpan bilangan bulat 32-bit, yang memiliki rentang nilai dari -2.147.483.648 hingga 2.147.483.647.

Contoh: val myInt: Int = 42

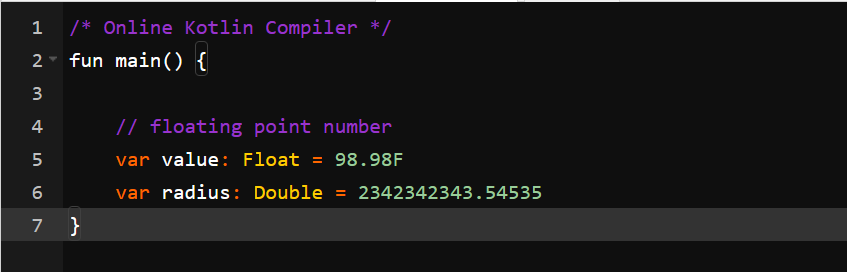
Long adalah tipe data yang digunakan untuk menyimpan bilangan bulat 64-bit, yang memiliki rentang nilai yang jauh lebih besar daripada Int. Rentang nilai Long adalah dari -9.223.372.036.854.775.808 hingga 9.223.372.036.854.775.807.

Kita harus menggunakan L atau l pada akhir nilai literal untuk menunjukkan bahwa itu adalah Long (misalnya, val myLong: Long = 10000000000L).

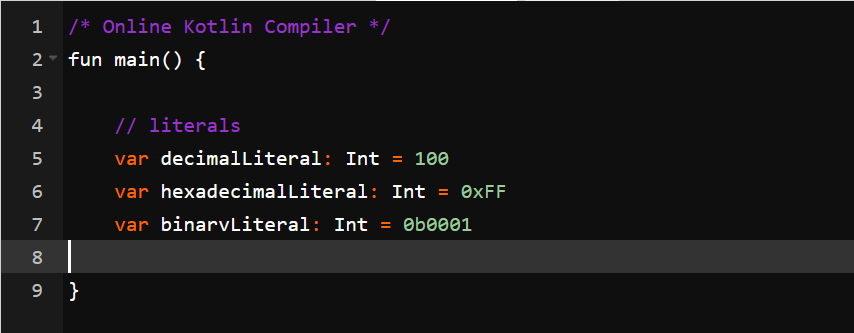


* **Floating Point Number**

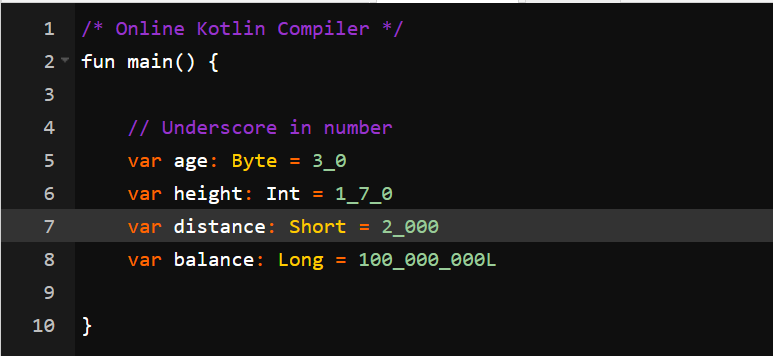
Dua tipe data floating-point yang umum digunakan dalam pemrograman adalah Float (32-bit) dan Double (64-bit). Tipe data Float digunakan untuk menyimpan angka dengan presisi lebih rendah tetapi dengan ukuran penyimpanan yang lebih kecil, sementara Double digunakan untuk menyimpan angka dengan presisi yang lebih tinggi dan rentang nilai yang lebih luas, meskipun dengan ukuran penyimpanan yang lebih besar.



* **Literals**

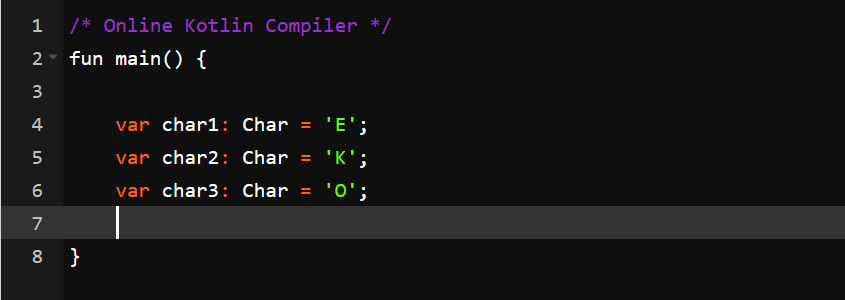


* **Underscore**



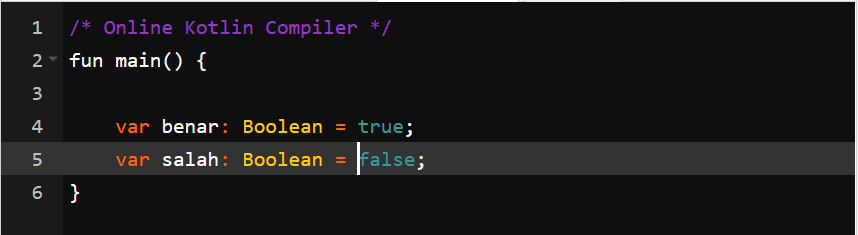
1. **Tipe Data Character**

Data karakter huruf, dikotlin dipersentasikan oleh tipe char. Untuk membuat data char, di kotlin kita bisa menggunakan tanda petik satu ( ‘ ).



1. **Tipe Data Boolean**

Tipe data Boolean adalah tipe data yang hanya memiliki 2 nilai yaitu benar atau salah. Tipe data Boolean di kotlin direpresentasikan dengan kata kunci Boolean. Nilai benar direpresentasikan dengan kunci true dan nilai salah direpresentasikan dengan kata kunci false.

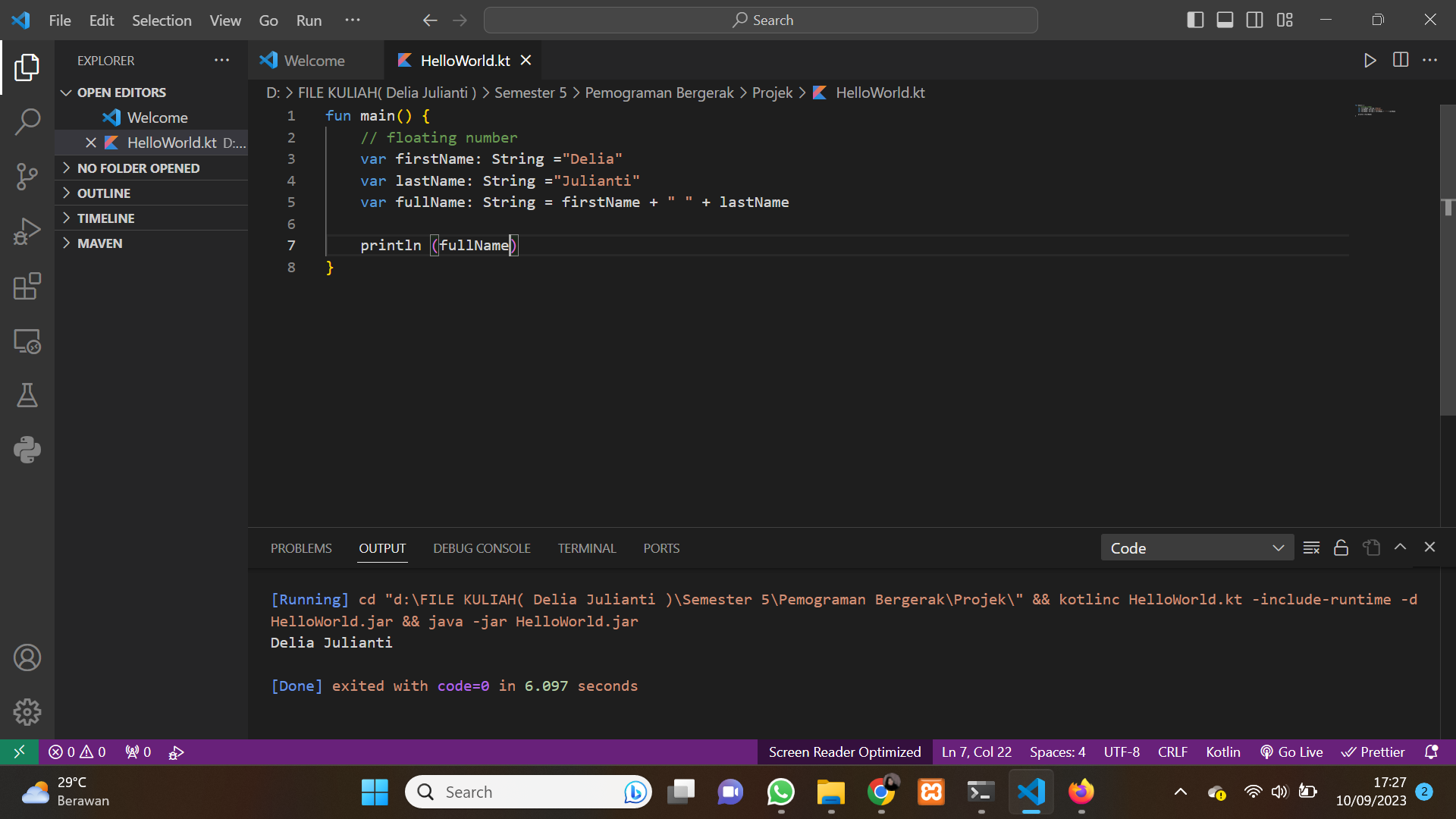


1. **Tipe data string**

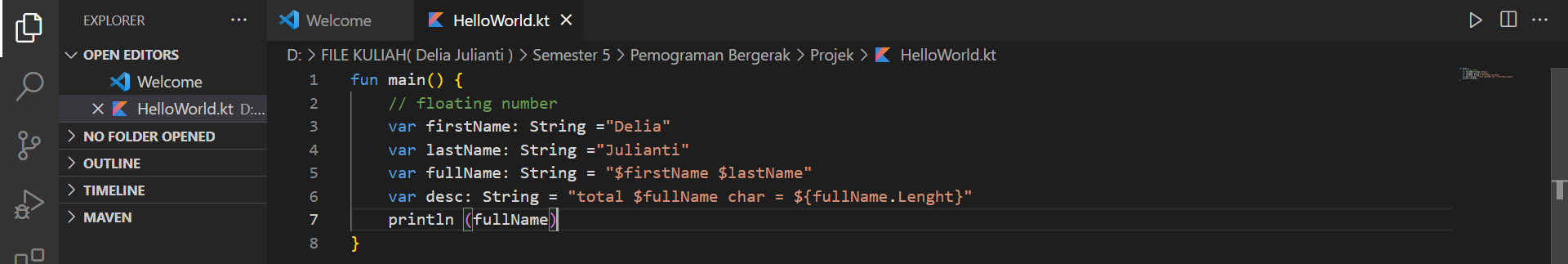
Tipe data string adalah tipe data yang berisikan data kumpulan karakter atau sederhananya adalah text. Di kotlin tipe data direpresentasikan dengan kata kunci string. Untuk membuat string dikotlin kita bisa menggunakan tanda petik 2 untuk string satu baris dan tanda petik 3 untuk text lebih dari satu baris.



* **Menggabungkan String**



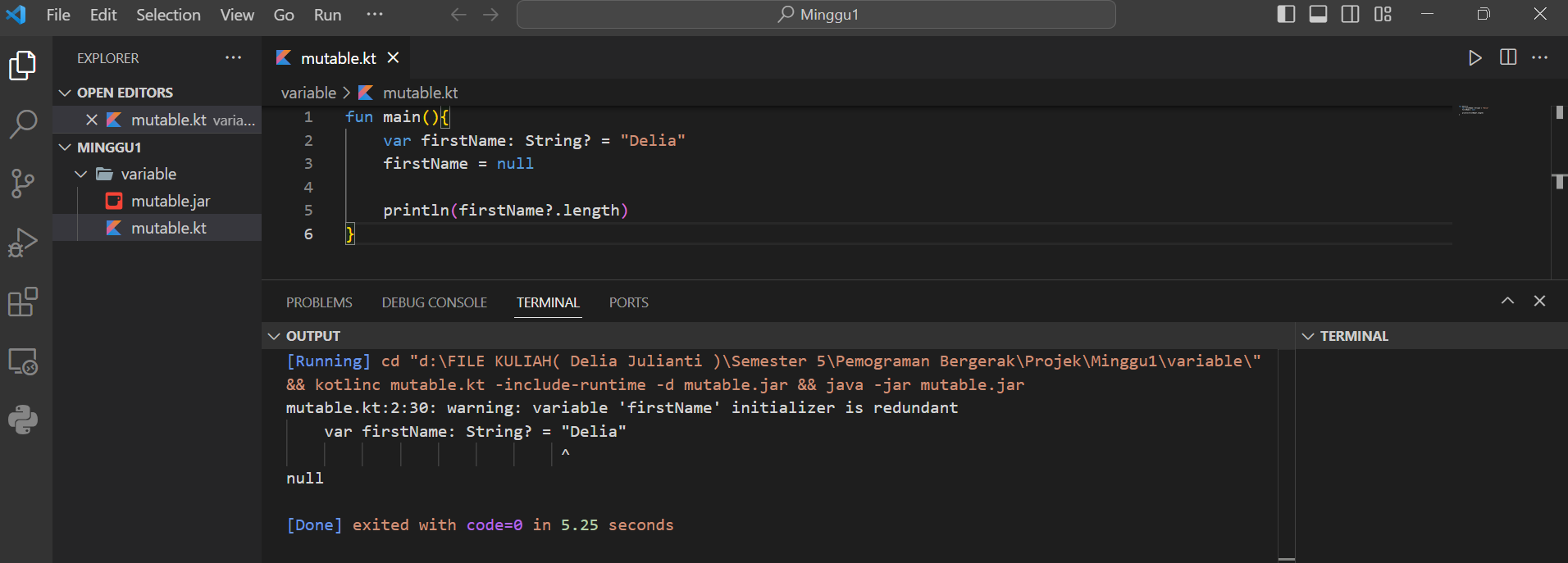
* **String Template**



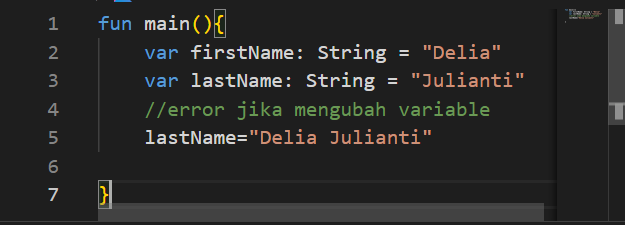
1. **Variable**

Variabel adalah tempat untuk menyimpan data. Kontlin mengandung 2 jenis variabel yaitu mutable (bisa diubah) dan immutable (tidak bisa diubah).

* Variable Mutable

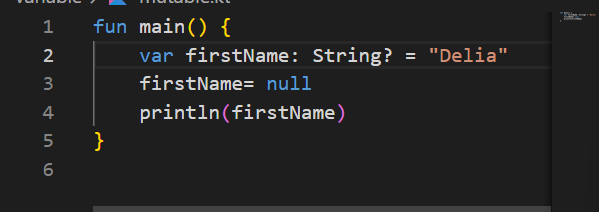


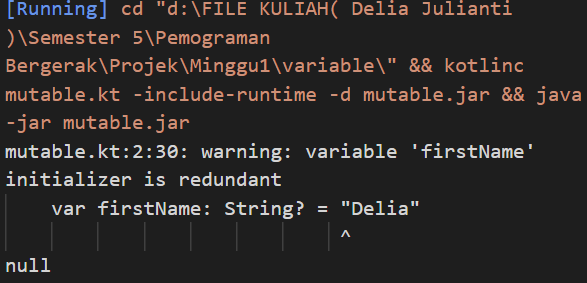
* **Variable Immutable**



* **Nullable**

Kotlin mendukung variable yang boleh null (tidak memiliki data) 0 Ini dikarenakan Kotlin bisa mengakses Java, dan kebanyakan di Java, semua variable bisa null. Untuk membuat variable bisa bernilai null, di Kotlin bisa menggunakan ? (tanda tanya) setelah tipe datanya. Penggunaan fitur ini tidak direkomendasikan untuk dilakukan di kotlin, hanya sebagai jalan akhir jika misal mengakses kode Java.





* **Variable constant**

Variabel konstan (constant variable) adalah sebuah variabel yang nilainya tidak dapat diubah setelah diberikan nilai awal pertama kali. Dalam berbagai bahasa pemrograman, termasuk Kotlin, variabel konstan dideklarasikan dengan kata kunci yang mengindikasikan bahwa nilainya tetap (konstan) setelah diberikan. Untuk menandai bahwa vaiabel itu constant, biasanya menggunakan UPPER\_CASE dalam pembuatan variabel constant nya

1. **Tipe Data Array**

Tipe data yang digunakan untuk menyimpan beberapa nilai sejenis dalam urutan tertentu. Tipe data Array dalam Kotlin bersifat mutable, yang berarti Kita dapat mengubah elemennya setelah inisialisasi. Terdapat dua cara umum untuk mendefinisikan array dalam Kotlin:

* **Array Bersifat Mutable**: Kita dapat menggunakan Array dengan menggunakan fungsi konstruktor arrayOf() atau Array() untuk membuat array bersifat mutable.

val angka = arrayOf(1, 2, 3, 4, 5)

angka[0] = 6 // Mengubah nilai elemen ke-0 menjadi 6

* **Array Bersifat Immutable**: Kita juga dapat menggunakan arrayOf() untuk membuat array yang bersifat immutable (tidak dapat diubah). Untuk array ini, elemen-elemennya tidak dapat diubah setelah inisialisasi.

val angka = listOf(1, 2, 3, 4, 5)

// angka[0] = 6 // Ini akan menghasilkan kesalahan karena array bersifat immutable

1. **Tipe Data Range**

Range adalah cara yang berguna untuk menghasilkan atau mengiterasi nilai-nilai dalam rentang tertentu tanpa harus secara eksplisit menuliskan semua nilai tersebut. Terdapat dua tipe range utama dalam Kotlin:

* **Closed Range (Rentang Tertutup**): Rentang ini mencakup semua nilai dari titik awal hingga titik akhir, termasuk titik awal dan akhir. Untuk membuat rentang tertutup kita dapat menggunakan ‘..’

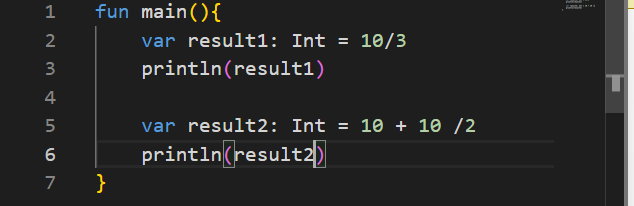
val range = 1..5 // Ini adalah rentang tertutup dari 1 hingga 5

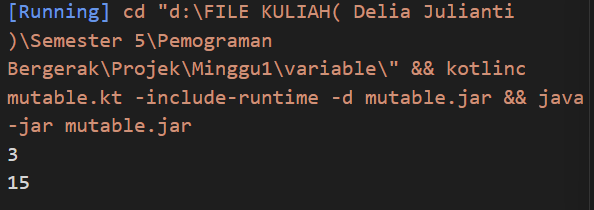
* **Half-Open Range (Rentang Separuh Terbuka):** Rentang ini mencakup semua nilai dari titik awal hingga sebelum titik akhir. Untuk membuat rentang separuh terbuka, Kita menggunakan operator until.

val range = 1 until 5 // Ini adalah rentang separuh terbuka dari 1 hingga sebelum 5

1. **Operasi Aritmatika**







* **Augmented assignment**

Adalah cara singkat untuk menggabungkan operator biner dengan operator assignment. Ini memungkinkan kita untuk melakukan operasi matematika atau bitwise pada variabel dan mengubah nilainya dengan cara yang lebih singkat.

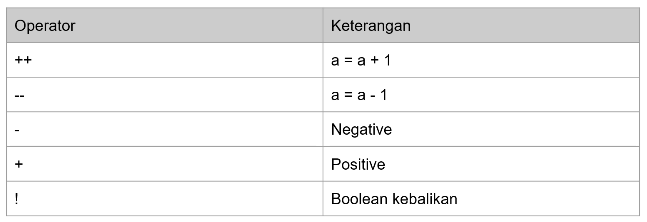
Contoh:

+= (Penambahan): x += 5 sama dengan x = x + 5

-= (Pengurangan): x -= 3 sama dengan x = x - 3

\*= (Perkalian): x \*= 2 sama dengan x = x \* 2

* **Unary operator**



1. **Operasi Perbandingan**

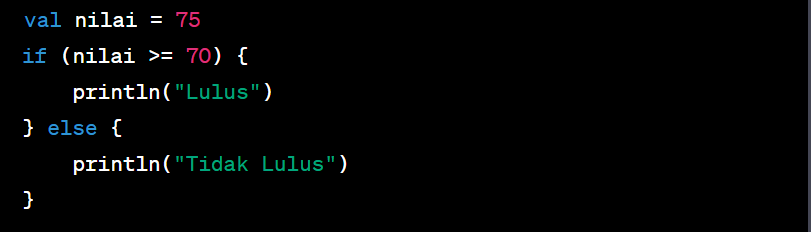


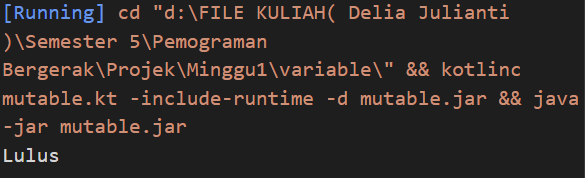
1. **Operasi Bolean**

* **Operasi Perbandingan:**
* == (sama dengan): Memeriksa apakah dua nilai sama.
* != (tidak sama dengan): Memeriksa apakah dua nilai berbeda.
  + (lebih besar dari): Memeriksa apakah nilai di sebelah kiri lebih besar dari nilai di sebelah kanan.
* < (lebih kecil dari): Memeriksa apakah nilai di sebelah kiri lebih kecil dari nilai di sebelah kanan.
* >= (lebih besar dari atau sama dengan): Memeriksa apakah nilai di sebelah kiri lebih besar dari atau sama dengan nilai di sebelah kanan.
* <= (lebih kecil dari atau sama dengan): Memeriksa apakah nilai di sebelah kiri lebih kecil dari atau sama dengan nilai di sebelah kanan.
* Operasi Logika:
* && (dan): Mengembalikan true jika kedua ekspresi boolean di kedua sisi operator && adalah true.
* || (atau): Mengembalikan true jika salah satu dari dua ekspresi boolean di kedua sisi operator || adalah true.
* ! (tidak): Menggantikan nilai boolean sehingga true menjadi false dan sebaliknya.

1. **If Expression**

if dapat digunakan sebagai ekspresi (expression if) atau sebagai pernyataan (statement if). Ini berarti kita dapat menggunakan if untuk mengembalikan nilai atau untuk mengontrol alur program. Berikut adalah penggunaan bentuk if:





1. **For Loops**

Dalam Bahasa kotlin, biasanya ada fitur yang Bernama perulngan yaitu for. For digunakan untuk melakukan perulangan iterasi dari data iterator (Array, Range, dan lainnya).

* Kode for array

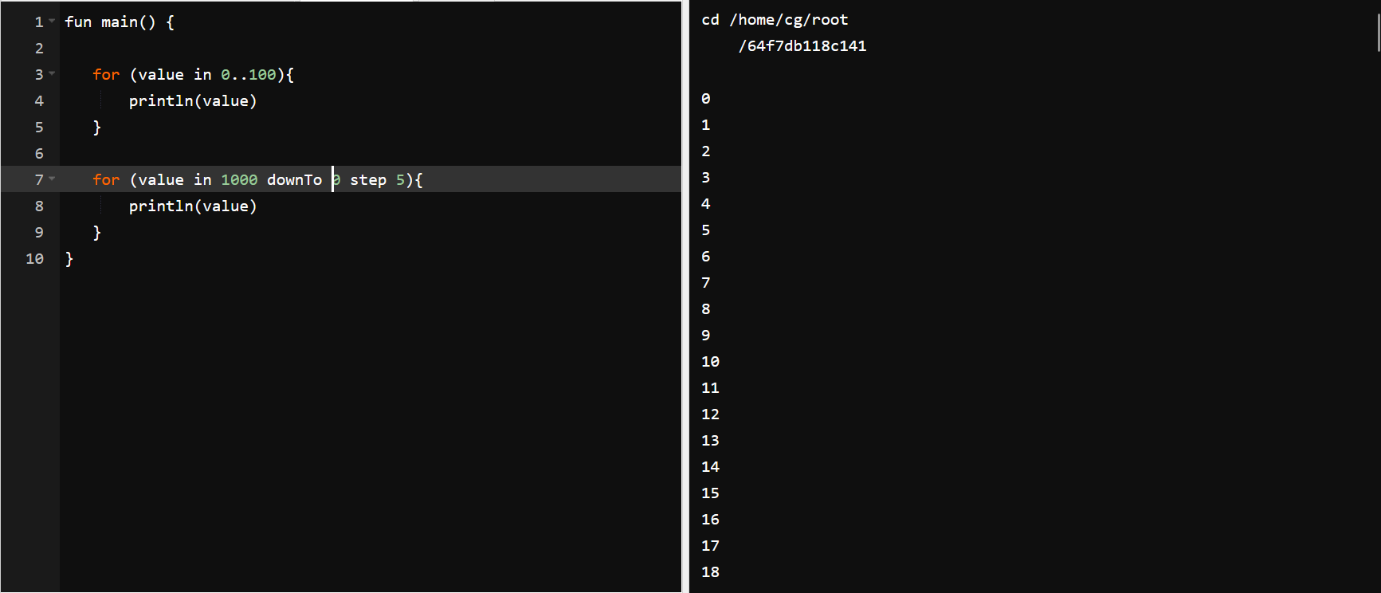
val angka = arrayOf(1, 2, 3, 4, 5)

for (item in angka) {

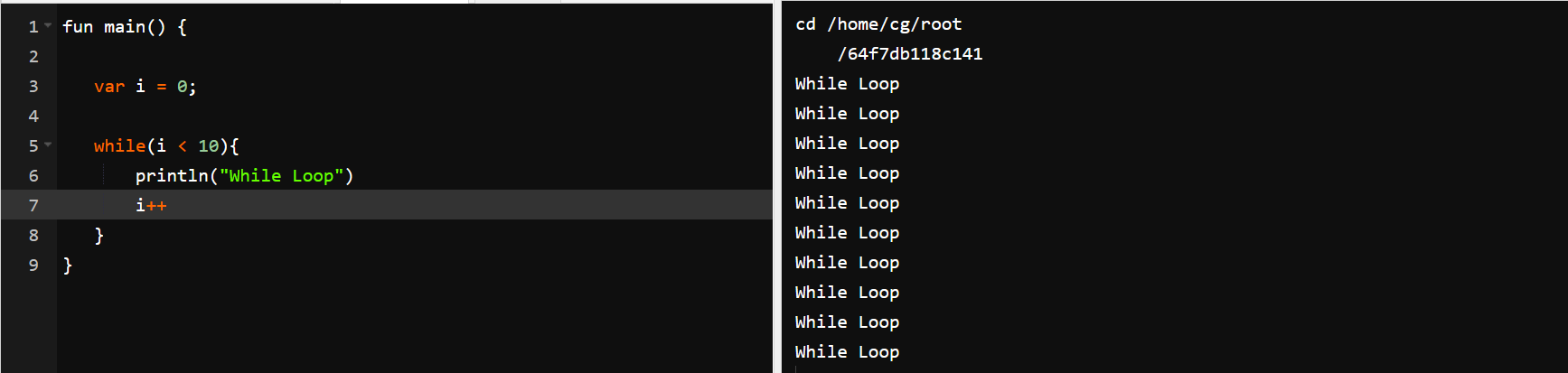
println(item)

}

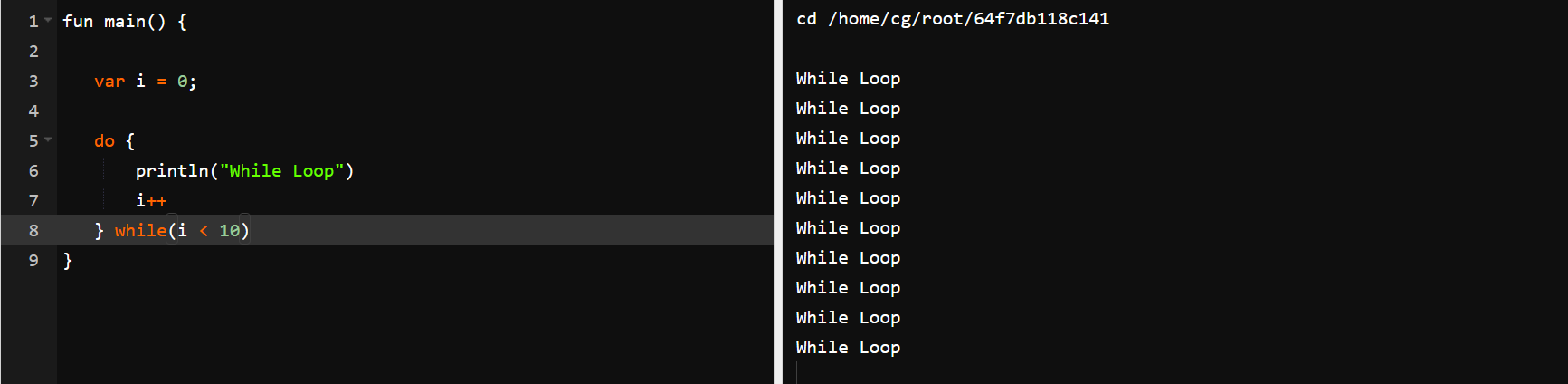
* Kode for range



1. **While Loops**

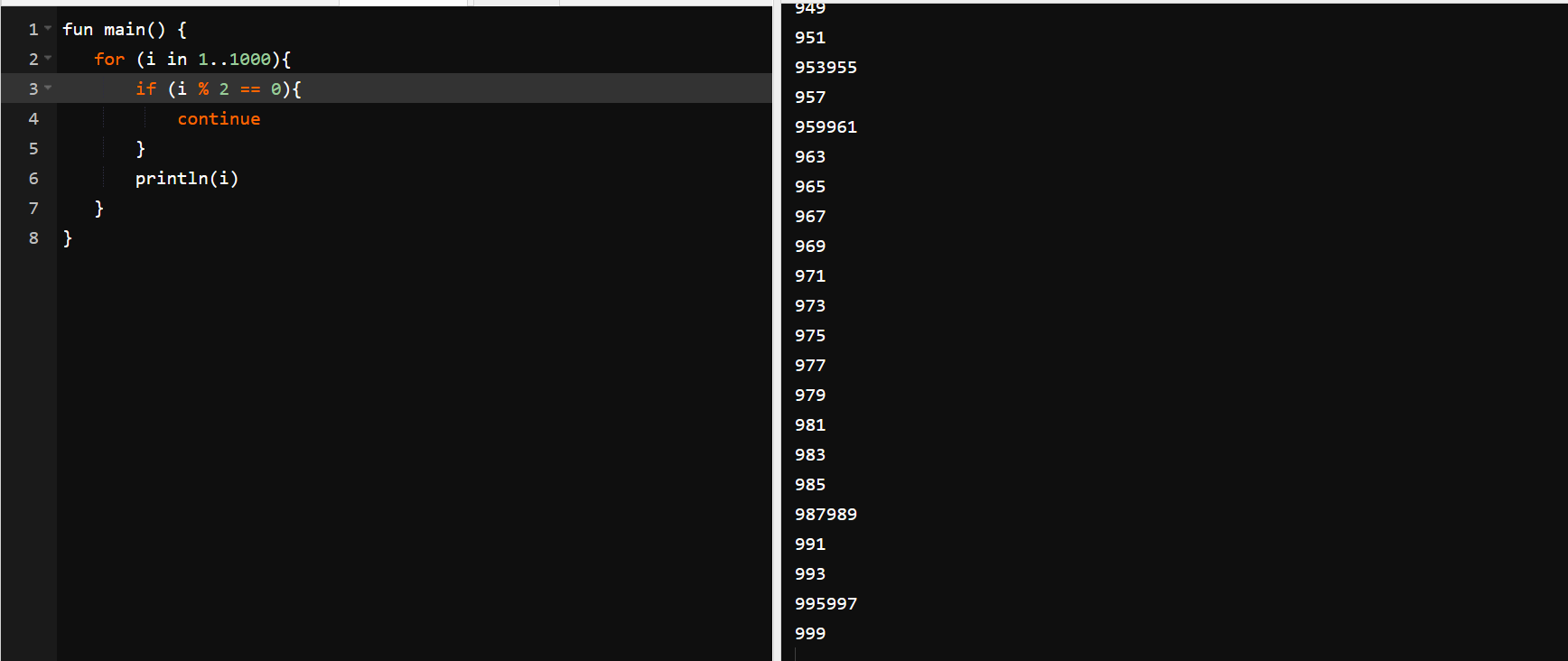
While adalah salah saru perulangan yang fleksible, dimana ode kotlin akan dilakukan pengecekan kondisi, jika kondisi bernilai true maka dia akan menjalankan blok while, dan terus diulangi samapai dalam kondisi while bernilai false. 

1. **Do While Loop**



1. **Break & Continue**

Break digunakan untuk menghentikan seluruh perulangan dan continue digunakan untuk menghentikan perulangan yang berjalan dan langsung melanjutkan ke perulangan selanjutnya.

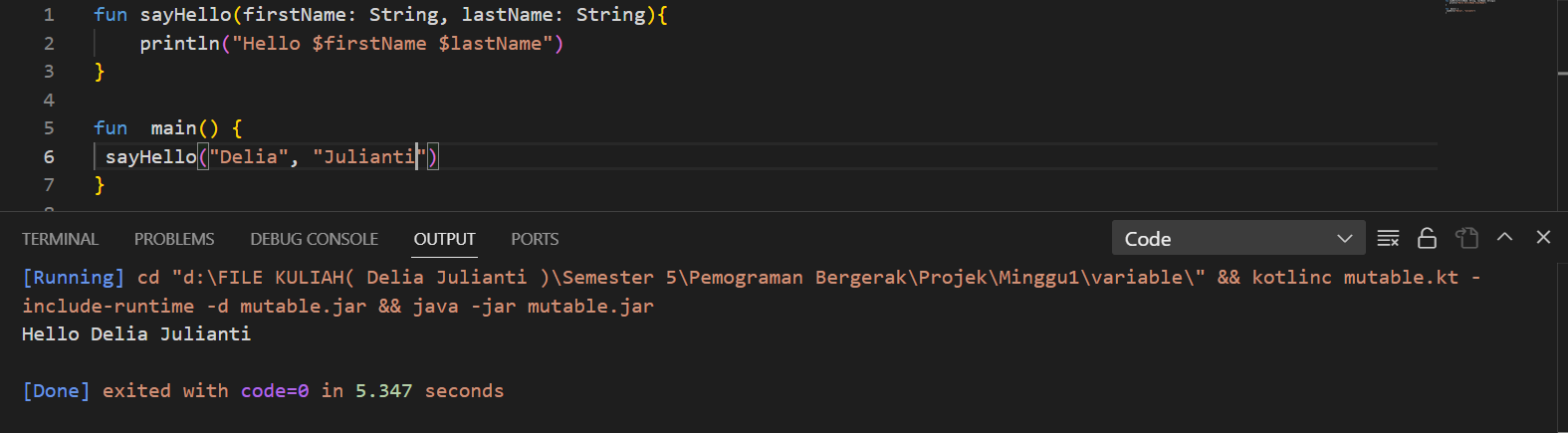


1. **Function**

Fungsi (functions) adalah bagian penting dalam Kotlin, seperti dalam bahasa pemrograman lainnya. Fungsi digunakan untuk mengelompokkan sejumlah pernyataan atau kode menjadi satu blok yang dapat dipanggil dengan nama tertentu. Ini membantu dalam mengorganisasi kode, menghindari duplikasi kode, dan membuat kode lebih mudah dibaca dan dimengerti.

* **Function Parameter**

Parameter adalah nilai yang diberikan ke fungsi saat fungsi tersebut dipanggil. Parameter memungkinkan kita untuk mengirim data ke dalam fungsi agar fungsi tersebut dapat melakukan tugas yang sesuai berdasarkan data yang diberikan.



* **Function default Argument**

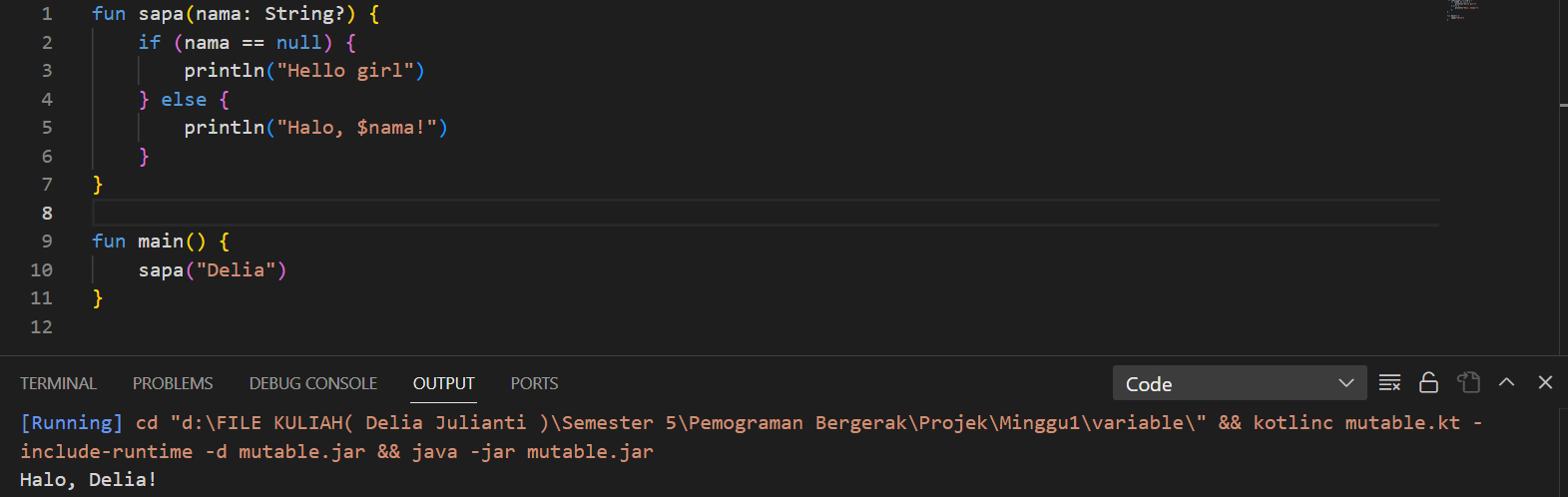
Fungsi dengan parameter default berguna ketika Kita ingin memberikan nilai default kepada parameter-parameter yang sering digunakan, tetapi masih memungkinkan penggunaan nilai yang berbeda jika diperlukan.

* **Function Named Argument**

Dalam Kotlin, Kita dapat menggunakan named argument (argumen bernama) saat memanggil fungsi. Named argument memungkinkan Kita untuk menyebutkan nama parameter saat memanggil fungsi, sehingga Kita dapat mengabaikan urutan parameter yang didefinisikan dalam fungsi tersebut.

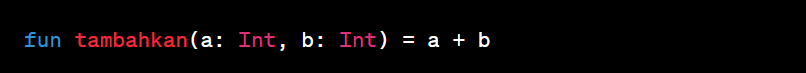
* **Unit Returning Function**

Fungsi yang mengembalikan tipe Unit disebut "unit returning function" dalam Kotlin. Tipe data Unit sebenarnya mirip dengan tipe void dalam bahasa pemrograman lain. Ini berarti fungsi tersebut tidak mengembalikan nilai yang bermakna atau tidak menghasilkan nilai yang harus disimpan atau digunakan oleh pemanggil fungsi.



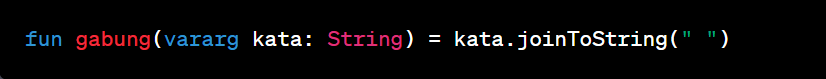
* **Single Expression Function**

Fungsi ini adalah fungsi yang memiliki hanya satu ekspresi sebagai isi fungsi. Karena hanya ada satu ekspresi, Kita tidak perlu menggunakan blok {}. Fungsi ini sering digunakan untuk fungsi yang sederhana.



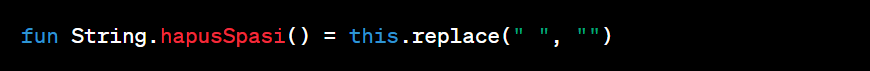
* **Functiob Varargs Parameter**

Parameter varargs adalah parameter dalam fungsi yang memungkinkan Kita untuk mengirimkan sejumlah argumen dari jenis yang sama tanpa harus menentukan jumlahnya secara eksplisit dalam deklarasi fungsi. Dalam fungsi, parameter varargs diwakili dengan vararg.



* **Extension Function**

Fungsi ekstensi adalah fitur unik Kotlin yang memungkinkan Kita menambahkan fungsi baru ke kelas yang sudah ada tanpa harus mengubah kode sumber kelas tersebut. Kita dapat memanggil fungsi ekstensi seperti metode biasa.



* **Function Infix Notation**

Notasi infix memungkinkan Kita untuk memanggil fungsi dengan operator yang mirip dengan operator matematika. Fungsi dengan notasi infix harus memiliki hanya satu parameter.

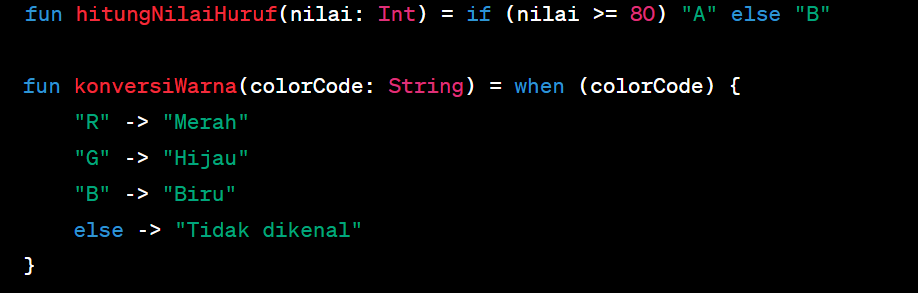


* **Function Scope**

Ini merujuk pada seberapa besar cakupan atau visibilitas sebuah fungsi. Dalam Kotlin, Kita dapat mendefinisikan fungsi di berbagai tingkatan cakupan seperti di dalam kelas, di dalam fungsi lain, atau bahkan di level modul.

* **Return if & When**

if dan when dapat digunakan sebagai ekspresi untuk mengembalikan nilai dari fungsi. Dalam beberapa kasus, Kita dapat menggantikan penggunaan blok {} dengan if atau when untuk mengembalikan nilai sesuai kondisi.



* **Recursive Function**

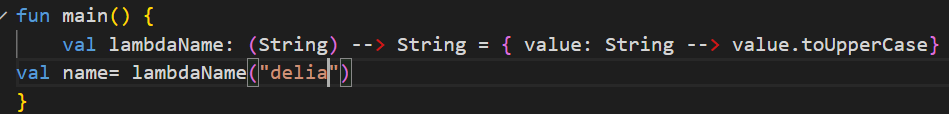
Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri selama eksekusi. Rekursi berguna dalam pemrosesan berulang atau dalam kasus-kasus di mana tugas yang sama perlu dilakukan berulang kali dengan variasi parameter tertentu.

* **Tail Recursive Funtion**

Fungsi rekursif ekor adalah jenis khusus dari fungsi rekursif di mana pemanggilan rekursif terakhir dalam fungsi. Dalam fungsi rekursif ekor, hasil dari pemanggilan rekursif langsung dikembalikan tanpa ada operasi tambahan.

* **Lambda Expression**

Lambda expression adalah cara singkat untuk mendefinisikan fungsi anonim (fungsi tanpa nama) dalam Kotlin. Lambda biasanya digunakan sebagai argumen untuk fungsi tingkat tinggi seperti map, filter, dan forEach, atau untuk menggantikan objek fungsi (fungsi sebagai objek) dalam pemrograman fungsional.



* **Higher-Order Funtion**

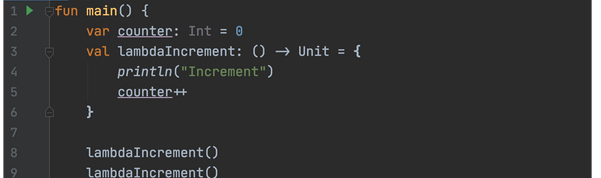
Fungsi tingkat tinggi adalah fungsi yang menerima satu atau lebih fungsi sebagai argumen atau mengembalikan fungsi sebagai hasil. Mereka memungkinkan Kita untuk memperlakukan fungsi sebagai objek dan menggabungkan pemrograman fungsional ke dalam kode Kita.

* **Anonymous Functions**

Fungsi anonim dalam Kotlin adalah fungsi tanpa nama yang dapat digunakan untuk menentukan blok kode yang dapat dieksekusi. Mereka umumnya digunakan sebagai argumen dalam pemanggilan fungsi tertentu, terutama dalam pemrograman fungsional. Fungsi anonim sering digunakan saat Kita perlu mendefinisikan logika yang sederhana dan tidak perlu membuat fungsi terpisah.

* **Closure**

Closure adalah konsep di mana fungsi dapat mengakses variabel dari lingkup luar fungsi tersebut. Dalam Kotlin, ini berarti bahwa fungsi anonim atau lambda expression dapat mengakses variabel yang didefinisikan di luar fungsi tersebut. Ini memungkinkan Kita untuk menjaga keadaan dan informasi di luar fungsi tetap tersedia untuk digunakan di dalamnya.

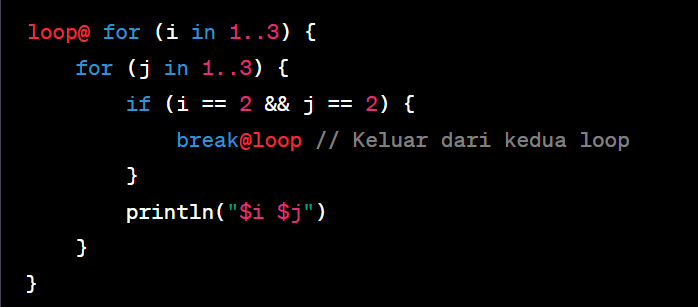


* **Inline Funtion**

Fungsi inline adalah jenis fungsi yang digunakan untuk menggantikan pemanggilan fungsi dengan kode sumber fungsi itu sendiri pada saat kompilasi. Hal ini berguna untuk menghindari overhead pemanggilan fungsi yang dapat mempengaruhi kinerja. Fungsi inline sangat berguna dalam konteks ekspresi lambda atau fungsi tingkat tinggi.

* **Label**

Label adalah penkita yang dapat ditempatkan di awal blok kode tertentu dalam Kotlin. Mereka digunakan bersamaan dengan pernyataan seperti `break`, `continue`, atau `return` untuk menentukan blok kode yang mana yang harus dikeluarkan atau dilewati. Label sangat berguna saat Kita memiliki loop bersarang dan ingin mengontrol perilaku keluar dari loop tertentu.

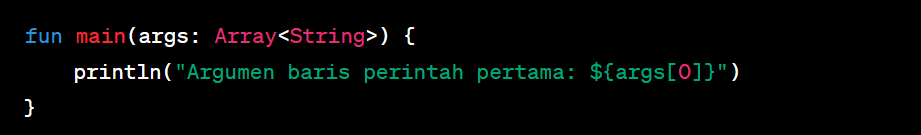


* **Package & Import**

Dalam Kotlin, kode Kita dapat diorganisir ke dalam paket (package) yang memungkinkan Kita mengelompokkan kelas dan fungsi berdasarkan tujuan atau fungsionalitasnya. Penggunaan `import` memungkinkan Kita untuk mengakses kelas dan fungsi dari paket lain tanpa harus menuliskan nama lengkapnya setiap kali. Ini membantu dalam pengorganisiran kode yang lebih baik dan penggunaan ulang kode.

* **Main Parameter**

Parameter `main` adalah cara di mana Kita dapat mengakses argumen baris perintah yang dilewatkan ke aplikasi Kotlin Kita saat menjalankannya dari baris perintah. Argumen ini tersedia dalam bentuk array dari string (`args: Array<String>`), dan mereka memungkinkan Kita untuk berinteraksi dengan pengguna atau mengkonfigurasi perilaku program Kita berdasarkan input eksternal.



* **Komentar**

Komentar adalah teks dalam kode yang tidak akan dieksekusi oleh komputer, tetapi digunakan untuk memberikan penjelasan, dokumentasi, atau informasi tambahan tentang kode tersebut. Dalam Kotlin, komentar dapat ditulis dengan dua bentuk: komentar satu baris yang dimulai dengan `//` dan komentar multi-baris yang diletakkan di antara `/\*` dan `\*/`. Komentar membantu programmer dan pengembang untuk memahami kode, menjelaskan tujuan, dan memelihara dokumentasi kode yang baik.

