ساختار کلی برنامه Go

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, World!")
}
```

حالا بریم خطبه‌خط توضیح بدیم:

package main

این دستور اجرایی نیست، بلکه تعریفه.

هر برنامه‌ای که می‌خوای بتونه اجرا بشه، باید با package main شروع بشه.

Go با دیدن package main می‌فهمه که این فایل نقطه شروع برنامه‌ست.

import "fmt"

اینم دستور اجرایی نیست، بلکه داره یه ابزار آماده رو وارد برنامه می‌کنه.

fmt یه پکیج (کتابخونه) توی Go هست که برای چاپ کردن متن روی صفحه استفاده می‌شه.

بدون این خط، نمی‌تونیم از fmt.Println استفاده کنیم.

func main() { ... }

این یه تعریف تابعه، نه دستور.

main اسم تابعیه که برنامه ازش شروع می‌شه.

Go موقع اجرای برنامه، فقط از این نقطه وارد می‌شه.

fmt.Println("Hello, World!")

این تنها دستور اجرایی واقعی در این برنامه است.

این خط به برنامه می‌گه: "این متن رو روی صفحه چاپ کن"

کد	نوع کد	توضیح
package main	تعریف	مشخص می‌کنه این برنامه قراره اجرا بشه
import "fmt"	وارد کردن ابزار	ابزار fmt رو برای چاپ کردن میاره
func main() { ... }	تعریف تابع	برنامه از اینجا شروع می‌شه
fmt.Println("Hello...")	دستور	یه دستور واقعی برای چاپ کردن روی صفحه

برنامه‌نویسی چیه؟

برنامه‌نویسی یعنی به کامپیوتر بگی "چه کاری انجام بده."
ما مرحله‌به‌مرحله بهش دستور می‌دیم:

1. یه سری اطلاعات اولیه (ورودی) بهش می‌دیم.
2. اون اطلاعات رو پردازش می‌کنه (مثلاً جمع، ضرب یا تبدیل انجام می‌ده).
3. بعد به ما نتیجه (خروجی) رو نشون می‌ده.

مثال 1: جمع دو عدد

ما دو عدد داریم: 4 و 7
می‌خوایم جمعشون کنیم و نتیجه رو چاپ کنیم.

```
package main

import "fmt"

func main() {
    fmt.Println("نتیجه جمع:", 4 + 7)
}
```

چی شد اینجا؟

- ورودی ما: عدد 4 و 7 (که توی کد نوشتیم)
- پردازش :جمع کردنشون
- خروجی :چاپ نتیجه با fmt.Println

مثال 2: تبدیل دما از سانتی‌گراد به فارنهایت

فرمول تبدیل:

```
F = (C × 1.8) + 32
```

فرض کنیم دمای ما 30 درجه‌ی سانتی‌گراد باشه:

```
package main

import "fmt"

func main() {
    fmt.Println("دما به فارنهایت:", (30.0 * 1.8) + 32)
}
```

اینجا هم:

- ورودی :عدد 30 (درجه سانتی‌گراد)
- پردازش :با فرمول تبدیلش کردیم به فارنهایت
- خروجی :نتیجه رو چاپ کردیم

مثال 3: محاسبه مساحت دایره

فرمول مساحت دایره اینه:

```
A = π * r * r
```

ما مقدار عدد π (پی) رو به‌صورت تقریبی 3.14 در نظر می‌گیریم.

```
package main

import "fmt"

func main() {
    fmt.Println("مساحت دایره:", 3.14 * 4.0 * 4.0)
}
```

نکته:

در این مرحله، ورودی رو توی خود کد می‌نویسیم (به جای اینکه از کاربر بخوایم چیزی وارد کنه) وقتی بیشتر پیشرفت کردی، یاد می‌گیری چطوری از کاربر سؤال بپرسی و از اون داده‌ها استفاده کنی.

بررسی چند نوع داده

int (عدد صحیح)

اعداد بدون اعشار

```
25
```

float64 (عدد اعشاری)

اعدادی که قسمت اعشاری دارن

```
1.75
```

string (متن)

برای نگهداری نوشته‌ها

```
“Iran”
```

عملیات‌هایی که روی این داده‌ها می‌تونیم انجام بدیم

روی gint و float64 می‌تونیم

عمل	توضیح	مثال
+	جمع	9 + 3
-	تفریق	9 - 3
*	ضرب	9 * 3
/	تقسیم	9 / 3

روی string فقط می‌تونیم

عمل	توضیح	مثال
+	اتصال (چسباندن متن‌ها به هم)	“Hello,” + “ world!”

مفهوم Operator وOperand

Operator (عملگر): علامتیه که یه کاری انجام می‌ده مثل + - * /

Operand (عملوند): چیزیه که روش اون کار انجام می‌شه (مثل عددها یا متغیرها)

مثال:

```
9 + 3
```

در اینجا + نقش operator (عملگر) و اعداد 9 و 3 نقش operand (عملوند) را دارند

ترتیب اجرای دستورات (execution order)

کدهایی که توی main می‌نویسیم، خط‌به‌خط و از بالا به پایین اجرا می‌شن. یعنی هر دستور تموم شه، بعدی اجرا می‌شه.

```
package main

import "fmt"

func main() {
    fmt.Println("شروع برنامه")

    fmt.Println("نام:", "Sara")
    fmt.Println("سن:", 20)
    fmt.Println("پس اندازه سارا متشکل از پس انداز نقدی و بانکی:", 310000 + 7100000)

    fmt.Println("پایان برنامه")
}
```

عملیات با بیشتر از دو عملوند

حالت اول :همه‌ی عملگرها از یک نوع باشند

در این حالت، مثلاً فقط از جمع استفاده می‌کنیم:

```
fmt.Println(2 + 3 + 4) // خروجی: 9
```

نحوه اجرا از چپ به راست:

2 + 3 -> 5

5 + 4 -> 9

حالت دوم :عملگرهای مختلف در یک عبارت

مثلاً از جمع و تفریق با هم استفاده کنیم:

```
fmt.Println(10 + 5 - 3) // خروجی: 12
```

نحوه اجرا از چپ به راست:

10 + 5 -> 15

15 – 3 -> 12

حالا برسیم به تقدم عملگرها

بعضی عملگرها اولویت بالاتری دارن، مثلاً:

در عملیات ریاضی بر روی اعداد (صحیح و اعشاری) * و / زودتر از + و – انجام میشن. یعنی ابتدا از چپ به راست عملیات * و / انجام می شود سپس بعد از ارزیابی مجدد عملیات جمع و تفریق انجام می شود

مثلا:

```
خروجی: fmt.Println(2 + 3 * 4) // 14
```

نحوه اجرا از چپ به راست:

3 * 4 -> 12

2 + 12 -> 14

استفاده از پرانتز برای تغییر اولویت

اگر بخوای اول جمع بشه بعد ضرب

```
خروجی: fmt.Println((2 + 3) * 4) // 20
```

نحوه اجرا از چپ به راست:

2 + 3 -> 5

5 * 4 -> 20

نتیجه‌گیری

عملگرها	اولویت
* /	1
+ -	2

نکته:

اگه شک داشتی یا می‌خوای ترتیب رو خودت مشخص کنی، از پرانتز استفاده کن.

عملیات ترکیبی (int با float)

اگه توی یه عبارت از int و float64 با هم استفاده کنیم، Go بهت خطا می‌ده، چون تایپ‌ها باید یکی باشن. پس باید یکی رو تبدیل کنیم:

```
fmt.Println(float64(10) + 3.5) // خروجی: 13.5
```

مفهوم متغیر و ثابت

فرض کن قصد داریم مساحت دو دایره با دو شعاع مختلف (شعاع 3 سانت و 4 سانت) رو حساب کنیم

با توجه به مثالی که قبلا برای محاسبه مساحت دایره داشتیم برنامه زیر این محاسباتو انجام میده

```
package main

import "fmt"

func main() {
    fmt.Println("مساحت دایره به شعاع 3 :", 3.14 * 3.0 * 3.0)
    fmt.Println("مساحت دایره به شعاع 4:", 3.14 * 4.0 * 4.0)
}
```

این کد به خوبی کار میکنه و نتیجه رو چاپ میکنه، اما یه ایرادی داره! فرض کن میخوایم مساحت رو با دقت بیشتری حساب کنیم، میدونیم که هر چقدر عدد Pi رو دقیق تر بنویسیم محسابه دقیق تر میشه.

عدد Pi با 10 رقم اعشار:

```
3.1415926535
```

از اونجایی که برنامه ای که نوشتیم برای دو دایره با شعاع مختلف داره مساحتو حساب میکنه پس باید عدد Pi رو در هر فرمول مساحت قرار بدیم. اما این خیلی جالب نیست! بهتر بود تنها یک دفعه این عدد رو در برنامه تعریف میکردیم و هر بار که خواستیم تغیرش بدیم مطمئن

باشیم تمام فرمول ها از عدد جدید در محاسبشون استفاده میکنن. این کارو میشه با متغیر انجام داد.

برای تعریف متغیر به این روش عمل میکنیم:

```
var pi float64 = 3.1415926535
```

تعریف متغیر دارای 4 بخش هست. به این ترتیب:

- کلمه var
- نام متغیر
- نوع
- مقدار

نکته: نام متغیر باید با حرف (یا کاراکتر _) شروع بشه، فقط شامل حروف، اعداد و _ باشه، و نباید با کلمات رزرو شده زبان یکی باشه (مثلا نمیشه متغیری داشته باشیم که اسمش type باشه، چون کلمه type جزو کلمه های رزرو شده در زبان GO هستش)

حالا اگه برنامه رو بخوایم با استفاده از متغیر بازنویسی کنیم به این شکل میشه:

```
package main

import "fmt"

func main() {
    var pi float64 = 3.1415926535
    fmt.Println("3 مساحت دایره به شعاع :", pi * 3.0 * 3.0)
    fmt.Println("4 مساحت دایره به شعاع:", pi * 4.0 * 4.0)
}
```

حالا فرض کن متوجه شدیم قصد داشتیم مساحت دایره ای به شعاع 2 رو در ابتدا حساب کنیم اما اومدیم برای دایره به شعاع 3 حساب کردیم. الان مجبوریم برای اصلاح برنامه دوبار مقدار 3.0 رو با مقدار 2.0 جایگزین کنیم. پس بهتر نیست برای شعاع هم بیایم متغیر در نظر بگیریم؟

به طور کلی در هر نقطه از برنامه اگر قصد داشتیم "مقداری" در محاسبات استفاده کنیم که احتمال داره اون مقدار به دلیلی در آینده یا در حین اجرای برنامه تغیر کنه بهتره از متغیر استفاده کنیم.

نسخه اصلاح شده برنامه بعد از استفاده از متغیر برای شعاع:

```
package main

import "fmt"

func main() {
    var pi float64 = 3.1415926535
    var radius1 float64 = 2.0
    var radius2 float64 = 4.0
    fmt.Println("مساحت دایره به شعاع 3 :", pi * radius1 * radius1)
    fmt.Println("مساحت دایره به شعاع 4:", pi * radius2 * radius2)
}
```

اما واقعا متغیر داره چیکار میکنه؟

متغیر یعنی بیایم از رم کامپیوتر یک فضا به اندازه مشخص (به واحد بایت) درخواست کنیم و مقدار رو در اون قرار بدیم. در برنامه قبل برای محاسبه دو دایره با دو شعاع مختلف دو فضای حافظه (متغیر) از رم درخواست کردیم. آیا واقعا نیاز بود دو فضای جداگانه در نظر بگیریم؟

باید این سوالو از خودمون بپرسیم! آیا بعد از اینکه مساحت دایره رو با اون شعاع محاسبه و نتیجه رو در خروجی چاپ کردیم باز هم نیاز داریم که بدونیم مقداری که داخل اون متغیر بود چی بود؟ در این مثال خاص خیر! پس دلیل نداره دوتا متغیر جدا در نظر بگیریم

میتونیم تنها یکبار متغیر radius تعریف کنیم محاسبه مساحت برای دایره اول رو انجام بدیم و نتیجه رو چاپ کنیم، سپس مقدار radius رو تغییر بدیم و محاسبه مساحت برای دایره دوم رو انجام بدیم و نتیجه رو چاپ کنیم. این کار باعث میشه در مصرف فضای رم صرفه جویی بشه، همچنین برنامه از نظر خوانایی بهتر بشه.

```
package main

import "fmt"

func main() {
    var pi float64 = 3.1415926535
    var radius float64 = 2.0
    fmt.Println("3 مساحت دایره به شعاع :", pi * radius * radius)
    radius = 4.0
    fmt.Println("4 مساحت دایره به شعاع:", pi * radius * radius)
}
```

نکته:

فقط در هنگام تعریف متغیر نیاز هست که از کلمه var استفاده کنیم و type اون رو مشخص کنیم. در هنگام قرار دادن مقدار جدید میشه به سادگی تنها مقدار جدید رو با استفاده از عملگر = در اون قرار داد

```
radius = 4.0
```

چرا از متغیر استفاده میکنیم؟

دلیل	توضیح
مقدار قابل تغییر	وقتی مقدار در طول اجرای برنامه ممکنه عوض بشه (مثل سن کاربر یا موجودی حساب)
استفاده برای ورودی‌ها	وقتی مقدار از کاربر یا محیط گرفته می‌شه و هنوز مشخص نیست
عملیات و پردازش	برای نگهداری نتیجه‌ی یک عملیات یا مرحله‌ای از پردازش
نگهداری داده موقت	وقتی به یه حافظه موقت نیاز داریم تا مقداری رو نگه داریم

به طور خلاصه متغیر یعنی مقداری که یک بار در رم ذخیره میشه و توسط یک نام قابل دسترسی هست و در حین اجرای برنامه یا در هنگام نوشتن برنامه تغیر میکنه.

تعریف بالارو برای دو متغیری که تعریف کردیم بررسی میکنیم.

برای متغیر radius:

- یک بار در رم ذخیره شده
- توسط نام radius قابل دسترس هست
- در حین اجرای برنامه تغییر کرده (توسط دستور radius = 4.0)

برای متغیر pi:

- یک بار در رم ذخیره شده
- توسط نام pi قابل دسترس هست
- در حین اجرای برنامه تغییر نکرده و قرار هم نیست هیچوقت تغییر کنه!

با توجه به بررسی ای که انجام دادیم متوجه شدیم مقدار pi هیچوقت در برنامه تغیر نکرده و قرار هم نیست در آینده تغییر کنه. بنابراین در اینجا متغیر بودن pi برای ما اهمیت نداره. ما صرفا میخوایم از تکرار کردن مقدار 3.1415926535 در دو محاسبه جلوگیری کنیم از طرف دیگه اینکه تونستیم برای مقدار 3.1415926535 یک اسم انتخاب کنیم باعث شده برنامه خواناتر بشه. این 2 ویژگی رو میتونیم با استفاده از مفهومی به اسم ثابت در زبان GO داشته باشیم. پس بهتره برنامه رو بازنویسی کنیم و pi رو به صورت ثابت تعریف کنیم.

```
package main

import "fmt"

func main() {
    const pi float64 = 3.1415926535
    var radius float64 = 2.0
    fmt.Println("3 مساحت دایره به شعاع :", pi * radius * radius)
    radius = 4.0
    fmt.Println("4 مساحت دایره به شعاع:", pi * radius * radius)
}
```

نکته قابل توجه این هست که مقدار ثابت در رم ذخیره نمیشه، و در هنگام کامپایل شدن برنامه اون مقدار در هر عملیاتی که از اون ثابت استفاده شده جایگزین میشه.

نحوه تعریف ثابت:

```
const pi float64 = 3.1415926535
```

تعریف ثابت دارای 4 بخش هست. به این ترتیب:

- کلمه const
- نام ثابت
- نوع
- مقدار

نکته: برای تعریف ثابت مشخص کردن نوع الزامی نیست. یعنی میشه به این شکل هم ثابت رو تعریف کرد

```
const pi = 3.1415926535
```

چرا از ثابت استفاده میکنیم؟

دلیل	توضیح
مقدارش تغییر نمی‌کنه	وقتی می‌خوای یه مقدار همیشه ثابت باشه (مثل عدد π یا نرخ تبدیل)
خوانایی بالا	نشون می‌ده این مقدار معنی‌دار و قطعی هست، مثلاً const taxRate= 0.09
بهینه‌سازی کامپایلر	چون مقدار در زمان کامپایل مشخصه، سریع‌تر اجرا می‌شه و حافظه مصرف نمی‌کنه
جلوگیری از اشتباه	از تغییر ناخواسته‌ی مقادیری که نباید تغییر کنن جلوگیری می‌کنه

جدول مقایسه ثابت و متغیر

ویژگی	ثابت (const)	متغیر (var)
قابل تغییر؟	نه	بله
نوع‌دهی اجباری؟	معمولا نه! مگر در شرایط خاص	بله
ذخیره درRAM ؟	نه	بله
زمان تعیین مقدار	زمان کامپایل	زمان اجرا
قابل استفاده با ورودی؟	نه	بله
خوانایی و اطمینان	بالا	بسته به استفاده

محاسبه مجموع دارایی دو برادر و یک خواهر

```
package main


import "fmt"


func main() {
    var (
        cashDeposit     int
        inBankDeposit   int
    )

    // برادر اول
    cashDeposit = 540000
    inBankDeposit = 3850000
    fmt.Println("دارایی برادر اول:", cashDeposit + inBankDeposit)

    // برادر دوم
    cashDeposit = 1280000
    inBankDeposit = 1450000
    fmt.Println("دارایی برادر دوم:", cashDeposit + inBankDeposit)

    // خواهر
    cashDeposit = 200000
    inBankDeposit = 5900000
    fmt.Println("دارایی خواهر:", cashDeposit + inBankDeposit)
}
```

در این مثال کاربرد و استفاده دو متغیر برای چند پردازش مختلف را به خوبی بررسی کردیم.
نکته دیگر در این مثال روش جدید برای تعریف دو متغیر با استفاده از یک دستور است، در اینجا چون دو متغیر از نظر معنایی به هم نزدیک هستند و هر دو برای نگهداری مقدار پس انداز (یکی به صورت نقدی و دیگری موجودی بانکی) می باشد با استفاده از دستور زیر آن ها را در کنارهم و تنها توسط یک دستور تعریف کردیم. در واقع در این حالت در استفاده از کلمه کلیدی var صرفه جویی کردیم

```
var (
    cashDeposit     int
    inBankDeposit   int
)
```

مفهوم تعریف متغیر بدون مقداردهی اولیه

گاهی اوقات متغیر رو تعریف می‌کنیم، ولی بعداً مقدار می‌دیم. در این حالت باید نوع متغیر رو مشخص کنیم

مثال: تبدیل دما از فارنهایت به سانتی‌گراد

```
package main

import "fmt"

func main() {
    var f float64           // متغیر بدون مقدار اولیه
    f = 98.6                // مقداردهی بعدی

    c := (f - 32) * 5 / 9
    fmt.Println("سانتی‌گراد:", c)
}
```

تعریف متغیر با مقدار اولیه و بدون type

اگه همون لحظه مقدار بدی، Go خودش بر اساس نوع مقداری که در اون متغیر قرار گرفته نوع رو تشخیص میده و اعلام کردن نوع متغیر الزامی نیست.

```
var name = "Ali"        // Go متوجه میشه که نوعش string هست
var age = 22            // نوعش int هست
```

استفاده از (Short Variable Declaration) :=

تو مثال قبل دیدیم که وقتی در هنگام تعریف متغیر مقدار دهی انجام بدیم دیگه نیازی نیست نوع رو مشخص کنیم، زبان GO به کمک ما اومده حتی این حالتو ساده تر کرده. تو این حالت دیگه نیاز نیست کلمه var در ابتدای تعریف متغیر بذاریم. میتونیم به روش زیر متغیر رو تعریف کنیم.

```
name := "Sara"
age := 30
height := 1.68// نوعش int هست
```

این روش فقط داخل تابع کار می‌کنه و خیلی خلاصه و کاربردیه.

جدول خلاصه ۳ روش تعریف متغیر

روش تعریف	کی استفاده میشه	مثال
var x int	وقتی نوع مهمه ولی هنوز مقدار ندادی	var count int
var x = value	وقتی مقدار داری و می‌خوای Go خودش نوع رو بفهمه	var name = "Ali"
x := value	وقتی داخل تابع هستی و خلاصه‌نویسی می‌خوای	age := 25

تمرین 1: ماشین حساب ساده

یک ماشین حساب ساده نیاز داریم که حاصل جمع، تفریق، ضرب و تقسیم دو عدد را محاسبه و در خروجی نشون بده.

نکته: برای دو عدد دو متغیر در نظر بگیرید و از تکرار کردن مقدار عددی در هر یک از عملیات بپرهیزید

تمرین 2: بررسی مفهوم تقدم عملگرها

نتیجه هر کدوم از عبارات زیر رو بدون استفاده از کامپیوتر روی کاغذ حساب کن، سپس با برنامه Go صحتش رو بررسی کن.

```
3 + 4 * 2
(3 + 4) * 2
10 / 2 + 5 * 2
10 / (2 + 5) * 2
```

تمرین 3: محاسبه محیط دایره

ثابت pi را برابر 3.14159 تعریف کن. سپس برنامه‌ای بنویس که شعاع دایره را از کاربر بگیرد و محیط آن را محاسبه کند

نکته: شعاع دایره عددی است اعشاری، مثلا: 1.5

فرمول محاسبه محیط دایره:

```
c = 2 * pi * r
```

تمرین 4: محاسبه شاخص توده بدنی (BMI)

برنامه‌ای بنویس که قد (به متر) و وزن (به کیلوگرم) کاربر رو دریافت کنه، شاخص توده بدنی (BMI)رو محاسبه کنه و چاپ کنه.

فرمول محاسبه BMI:

```
bmi = weight / height * height
```

نکته: برای قد و وزن متغیر در نظر بگیرید و از قرار دادن مستقیم مقدار عددی در فرمول بپرهیزید. متغیر قد از نوع اعشاری و متغیر وزن از نوع عدد صحیح است.

نکته: برای انجام محاسبه ترکیبی (انجام عملیات ریاضی بر روی متغیر از نوع صحیح و اعشاری در یک دستور) نیاز هست یکی از آن ها تبدیل شوند. مثلا متغیر وزن که از نوع عدد صحیح است تبدیل به اعشار شود.