

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 1
MODUL 13
“REPEAT-UNTIL”



DISUSUN OLEH:
ANASTASIA ADINDA NARENDRA INDRIANTO
103112400085
S1 IF-12-01

DOSEN:
Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024/2025

DASAR TEORI

1. Konsep Input dan Output

Input merupakan intruksi dasar untuk membaca data yang diberikan dari pengguna. Data yang diberikan oleh pengguna akan disimpan ke dalam suatu wadah yang disebut *variable*. Penulisan intruksi input beragam menyesuaikan Bahasa pemrograman yang digunakan. Output merupakan perintah untuk menampilkan data ke layar monitor. Data yang sudah diproses atau diolah oleh program komputer perlu ditampilkan ke layar sehingga pengguna bisa memperoleh informasi dari hasil pengolahan data yang dilakukan program.

2. Konsep Data, Variabel, dan Intruksi Dasar

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan. Nama variabel dimulai dengan huruf dan dapat diikuti dengan sejumlah huruf, angka atau garisbawah. Tipe data yang umum adalah integer, real, Boolean, karakter dan string. Nilai data yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnya. Informasi Alamat atau Lokasi variabel dapat diperoleh dengan menambahkan prefix dan di depan nama variabel tersebut.

3. Konsep Bahasa Pemrograman Go

Bahasa Go menganut kesesuaian tipe data yang ketat. Tipe data yang berbeda tidak boleh dicampur dalam satu ekspresi, bahkan tipe data masih yang sejenis. Menyesuaikan tipe data ada beberapa cara yaitu *casting tipe (data)* mengubah tipe dari data yang diberikan ke tipe data yang diinginkan, memanfaatkan fungsi *Sprint* dan *Sscan* dari paket *fmt*, dan memanfaatkan fungsi-fungsi dalam paket *strconv*, seperti *Atoi*, *Itoa* dan *ParseBool*. Variabel harus dideklarasikan terlebih dahulu sebelum digunakan. Variabel juga harus diinisialisasi dulu agar nilai yang tersimpan diketahui dengan jelas dan eksekusi algoritma menjadi terprediksi. Dalam Bahasa Go, variabel yang tidak diinisialisasi lebih dahulu otomatis diisi dengan nilai default ekuivalen dengan bit 0.

4. Konsep Tipe Data

Melalui ilmu komputer tentunya kita sering mendengar istilah data. Data terbagi menjadi bermacam-macam tipe data yang terklasifikasi dan memiliki fungsi sendiri. Tentunya, berbagai macam tipe data tersebut sangat bermanfaat bagi kinerja komputer melalui kode-kode dalam bahasa pemrograman. Data types atau tipe data adalah sebuah pengklasifikasian data berdasarkan jenis data tersebut. Tipe data dibutuhkan agar kompiler dapat mengetahui bagaimana sebuah data akan digunakan. Untuk mengembangkan sebuah program ada beberapa tipe data yang dibutuhkan. Tipe data memiliki 4 data type yang sering digunakan yaitu,

- 1) Bilangan Bulat (Integer) Tipe bilangan bulat (Integer) adalah tipe data numerik yang biasa digunakan apabila bertemu dengan bilangan bulat, seperti 1, 27, 100, dll. Bilangan ini juga mengenal nilai positif dan negatif. Tipe data numerik yang termasuk ke dalam bilangan bulat adalah *byte*, *short*, *int*, dan *long*.
- 2) Bilangan Pecahan (Floating Point) Tipe bilangan pecahan atau *floating point* adalah bilangan yang menangani bilangan desimal atau perhitungan secara detail. Karena kemampuannya, float point berbanding terbalik dengan integer. Terdapat dua tipe pada bilangan pecahan ini yaitu *float* dan *double*.
- 3) Karakter (Char)
Tipe data karakter tunggal yang biasa didefinisikan dengan tanda petik (') di awal dan di akhir karakternya. Tipe ini mengikuti aturan “unicode” sehingga bilangan harus

diawali kode “/u”. Tetapi juga biasa menggunakan bilangan heksadesimal dari 0000 sampai FFFF.

4) Boolean

Tipe data *boolean* merupakan tipe yang memiliki dua nilai yaitu benar (*true*) atau salah (*false*). Nilai yang digunakan pada tipe ini sangat penting dalam mengambil keputusan suatu kejadian tertentu.

5. **Deklarasi dan Inisialisasi**

Deklarasi variabel dalam bahasa pemrograman komputer adalah pernyataan yang digunakan untuk menentukan nama variabel dan tipe datanya. Deklarasi memberi tahu kompiler tentang keberadaan entitas dalam program dan lokasinya. Saat Anda mendeklarasikan variabel, Anda juga harus menginisialisasinya. Inisialisasi adalah proses pemberian nilai pada Variabel. Setiap bahasa pemrograman memiliki metode tersendiri untuk menginisialisasi variabel. Jika nilai tidak diberikan pada Variabel, maka proses tersebut hanya disebut Deklarasi.

6. **Konstanta dalam Bahasa Pemrograman**

Konstanta adalah entitas yang merujuk pada nilai data yang tetap dan tidak dapat diubah. Selama eksekusi/perhitungan pemrograman, nilai suatu konstanta tidak dapat diubah, tetap konstan. Jenis Konstanta ada konstanta Bilangan Bulat. Konstanta bilangan bulat adalah barisan bilangan bulat yang nilainya tetap. Mereka tidak boleh mengandung koma desimal atau angka pecahan. Konstanta bilangan bulat dapat berupa bilangan positif atau negatif. Mereka termasuk bilangan bulat sistem desimal, bilangan bulat sistem oktal, bilangan bulat sistem heksadesimal.

7. **Integer Division dan Modulo**

Pembagian pada tipe data integer (**integer division** atau **div**) sedikit berbeda dengan pembagian yang sudah sering kita pelajari dari sekolah dasar. Hasil pembagian akan bertipe data integer, artinya kita akan mengabaikan bilangan yang muncul setelah tanda koma atau floating point. Hasil dari operasi div ini dikenal dengan istilah **quotient**. Modulo atau modulus "mod" adalah operasi untuk mencari nilai integer dari sisa pembagian pada integer division, biasanya dikenal juga dengan istilah remainder. Misalnya untuk operasi 10 dibagi dengan 3, maka sisa pembagian adalah 1. Operator modulo menggunakan "%" di dalam bahasa pemrograman Go.

8. **Casting atau Konversi Tipe Data**

Pada Bahasa pemrograman Go, tipe data bersifat statis, artinya tipe data yang sudah didefinisikan tidak dapat diganti selama program berjalan. Casting merupakan salah satu Teknik konversi antar tipe data di dalam Bahasa pemrograman.

9. **Paradigma Perulangan**

Perulangan merupakan salah satu struktur kontrol yang memungkinkan suatu instruksi yang sama dilakukan berulang kali dalam waktu atau jumlah yang lama. Tanpa instruksi perulangan, maka suatu instruksi akan ditulis dalam jumlah yang sangat banyak. Salah satu instruksi perulangan yang paling mudah adalah **for-loop**, yang mana dengan instruksi ini dapat digunakan untuk mengulangi instruksi sebanyak **n** kali (iterasi). Batasan besar nilai dari **n** menyesuaikan dengan batasan dari tipe data integer yang digunakan.

10. **Paradigma Percabangan if-else**

Percabangan if-else adalah cara untuk menjalankan kode yang berbeda tergantung pada apakah kondisi yang diberikan bernilai benar (*true*) atau salah (*false*). Bentuk paling sederhana dari percabangan ini adalah dengan menggunakan if untuk memeriksa kondisi, artinya kode program dari baris ke-1 hingga baris terakhir akan dieksekusi satu persatu.

11. Pengertian Switch-Case

Switch case adalah pernyataan pilihan ganda dalam bahasa pemrograman yang memungkinkan nilai variabel atau ekspresi untuk mengubah aliran kontrol eksekusi program. Penggunaan if-else dan switch-case bergantung pada kebutuhan Anda. if-else digunakan ketika Anda memiliki beberapa kondisi yang berbeda, sementara switch-case digunakan ketika Anda memiliki banyak pilihan yang mungkin dengan variabel yang sama.

12. Karakteristik Switch-Case

Pada dasarnya switch-case merupakan alternatif dari penggunaan else-if, tetapi pemilihan aksi tidak dilakukan berdasarkan kondisi, tetapi suatu nilai atau operasi tipe data dasar yang tidak menghasilkan boolean.

Penulisan switch case terdiri dari komponen berikut:

- a. Ekspresi, merupakan operasi tipe data yang menghasilkan nilai selain tipe data boolean,
- b. Nilai, pilihan case dari ekspresi yang dideklarasikan pada instruksi switch-case.
- c. Aksi, merupakan kumpulan instruksi yang akan dieksekusi sesuai dengan nilai yang dihasilkan dari ekspresi yang terdapat pada switch-case. Aksi lain hanya akan dieksekusi apabila tidak ada pilihan nilai yang sesuai dengan hasil ekspresi.

14. Paradigma Perulangan

Perulangan merupakan salah satu struktur kontrol yang memungkinkan suatu instruksi yang sama dilakukan berulang kali dalam waktu atau jumlah yang lama. Tanpa instruksi perulangan, maka suatu instruksi akan ditulis dalam jumlah yang sangat banyak. Sebelumnya pada modul ke-5 dan 6 telah dipelajari instruksi perulangan dengan for-loop. Instruksi for-loop memungkinkan kita melakukan berulangan sebanyak n iterasi, akan tetapi pada banyak kasus yang melibatkan perulangan, tidak semua perulangan diketahui jumlah iterasinya di awal. Perulangan seperti ini disebut juga dengan istilah perulangan dengan kondisi.

13. Karakteristik While-Loop

Struktur kontrol perulangan menggunakan while-loop memiliki bentuk yang hampir serupa dengan penulisan if-then pada percabangan, yaitu memiliki kondisi dan aksi. Hal yang membedakan adalah aksi akan dilakukan secara berulang-ulang selama kondisi bernilai true.

- 1) Kondisi, merupakan nilai atau operasi tipe data yang menghasilkan tipe data boolean.

Kondisi ini merupakan syarat terjadinya perulangan. Artinya perulangan terjadi apabila kondisi bernilai true.

- 2) Aksi, merupakan kumpulan instruksi yang akan dieksekusi secara berulang-ulang selama kondisi bernilai true. *Salah satu instruksi dari aksi harus bisa membuat kondisi yang awalnya bernilai true menjadi false, tujuannya adalah untuk membuat perulangan*

berhenti. Pada penulisan notasinya secara umum bahasa pemrograman menggunakan kata kunci while, tetapi khusus di bahasa pemrograman Go, kata kunci yang digunakan adalah for. Walaupun berbeda dari kata kunci yang digunakan, secara struktur penulisannya tetap sama, sehingga tetap mudah untuk membedakan instruksi for yang digunakan adalah for-loop atau while-loop

14. Pengertian Repeat-Until

Penggunaan repeat-until pada dasarnya sama dengan while-loop di mana perulangan berdasarkan kondisi. Perbedaan terletak pada kondisi yang digunakan, pada while-loop kondisi yang harus didefinisikan adalah kondisi perulangannya, atau kapan perulangan itu terjadi, sedangkan pada repeat-until kondisi yang harus didefinisikan merupakan kondisi berhenti, atau kapan perulangan tersebut harus dihentikan.

15. Karakteristik Repeat-Until

Komponen dari repeat-until sama dengan while-loop, yaitu terdapat kondisi dan aksi, hanya struktur penulisannya saja yang berbeda.

- 1) Aksi, merupakan kumpulan instruksi yang akan dilakukan perulangan. Aksi minimal dijalankan sekali, baru dilakukan pengecekan kondisi berhenti setelahnya. Apabila kondisi bernilai true, maka perulangan dihentikan.
- 2) Kondisi/berhenti, merupakan kondisi berhenti dari perulangan, harus bernilai false selama perulangan dilakukan.

CONTOH SOAL

1. Contoh Latihan Soal 1

Source Code:

```
package main

import "fmt"

func main() {
    var word string
    var repetitions int
    fmt.Scan(&word, &repetitions)
    counter := 0
    for done := false; !done; {
        fmt.Println(word)
        counter++
        done = (counter >= repetitions)
    }
}
```

Output:

```
n "d:\Semester1\Semester 1 AlPro\Go\MODUL13\lcoso.
go"
pagi 3
pagi
pagi
pagi
PS D:\Semester1\Semester 1 AlPro\Go\MODUL13> go run "d:\Semester1\Semester 1 AlPro\Go\MODUL13\lcoso.go"
kursi 5
kursi
kursi
kursi
kursi
kursi
```

Deskripsi Program:

Program lcoso.go digunakan untuk menerima input kata dan mencetaknya sebanyak jumlah pengulangan yang diinginkan oleh pengguna. Program akan dihentikan ketika jumlah kata yang dicetak mencapai jumlah yang diinginkan oleh pengguna.

Masukan berupa suatu kata dan jumlah pengulangan yang diinginkan oleh pengguna. berupa **Keluaran** kata yang diinputkan pengguna dan dicetak sebanyak jumlah pengulangan yang diinginkan oleh pengguna. User dapat menginputkan data yang diinginkan didalam bagian terminal setelah program dijalankan dengan runner program. Berikut input dan ouput;

No	Masukan	Keluaran
1	pagi 3	pagi pagi pagi
2	kursi 5	kursi kursi kursi kursi kursi

2. Contoh Latihan Soal 2

Source Code:

```
package main

import "fmt"

func main() {
    var number int
    var countinuloop bool
    for countinuloop = true; countinuloop; {
        fmt.Scan(&number)
        countinuloop = number <= 0
    }
    fmt.Printf("%d adalah bilangan bulat positif\n", number)
}
```

Output:

```
PS D:\Semester1\Semester 1 AIPro\Go\MODUL13> go ru
go"
-5
-2
-1
0
5
5 adalah bilangan bulat positif
PS D:\Semester1\Semester 1 AIPro\Go\MODUL13> go run "d:\Semester1\Semester 1 AIPro\Go\MODUL13\2coso.go"
17
17 adalah bilangan bulat positif
```

Deskripsi Program:

Program 2coso.go digunakan meminta pengguna untuk memasukkan bilangan bulat positif. Program akan terus meminta input hingga pengguna memasukkan bilangan bulat positif. **Masukan** berupa bilangan bulat positif, apabila bukan maka program akan terus meminta masukan hingga bilangan yang diberikan adalah bilangan bulat positif.

Keluaran berupa satu baris keluaran yang menunjukkan n bilangan adalah bilangan bulat positif. User dapat menginputkan data yang diinginkan didalam bagian terminal setelah program dijalankan dengan runner program. Berikut input dan ouput;

No	Masukan	Keluaran
1	-5 -2 -1 0 5	5 adalah bilangan bulat positif
2	17	17 adalah bilangan bulat positif

3. Contoh Latihan Soal 3

Source Code:

```
package main

import "fmt"

func main() {
    var N, s1, s2, j, temp int
    fmt.Scan(&N)

    s1 = 0
    s2 = 1
    j = 0
    for j < N {
        fmt.Print(s1, " ")
        temp = s1 + s2
        s1 = s2
        s2 = temp
        j = j + 1
    }
}
```

Output:

```
PS D:\Semester1\Semester 1 A1Pro\Go\MODUL13> go run "d:\Semester1\Semester 1 A1Pro\Go\MODUL13\3coso.go"
5
2
3
1
-1
false
PS D:\Semester1\Semester 1 A1Pro\Go\MODUL13> go run "d:\Semester1\Semester 1 A1Pro\Go\MODUL13\3coso.go"
15
3
12
9
6
3
0
true
PS D:\Semester1\Semester 1 A1Pro\Go\MODUL13> go run "d:\Semester1\Semester 1 A1Pro\Go\MODUL13\3coso.go"
25
5
20
15
10
5
0
true
```


Deskripsi Program:

Program 3coso.go digunakan melakukan pengecekan apakah suatu bilangan merupakan kelipatan dari bilangan lainnya.

Masukan terdiri dari dua buah bilangan bulat positif X dan Y.

Keluaran terdiri dari perulangan pengurangan kelipatan dengan hasil akhir boolean yang menyatakan apakah bilangan X merupakan kelipatan dari Y. User dapat menginputkan data yang diinginkan didalam bagian terminal setelah program dijalankan dengan runner program. Berikut input dan ouput;

No	Masukan	Keluaran
1	5 2	3 1 -1 false
2	15 3	12 9 6 3 0 true
3	25 5	20 15 10 5 0 true

SOAL LATIHAN

1. Latihan Soal 1

Source Code:

```
package main

import "fmt"

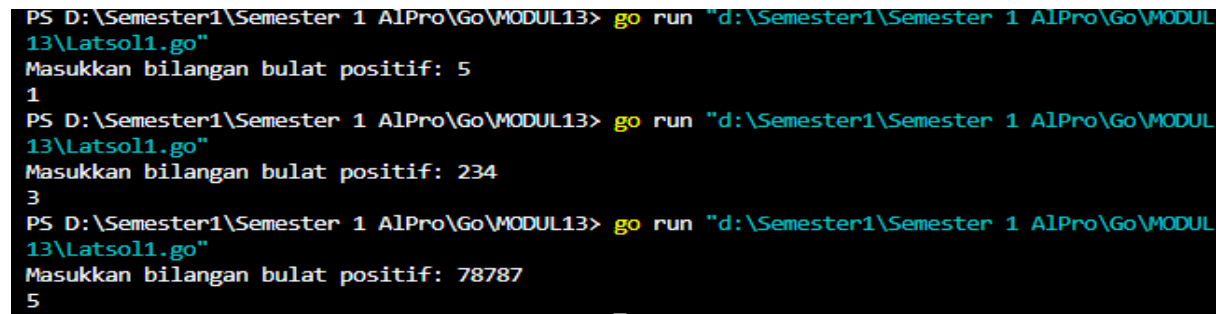
func main() {
    var input int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scanln(&input)

    if input <= 0 {
        fmt.Println("Masukan harus berupa bilangan bulat positif.")
        return
    }

    count := 0
    for input > 0 {
        input /= 10
        count++
    }

    fmt.Print(count)
}
```

Output:



```
PS D:\Semester1\Semester 1 APro\Go\MODUL13> go run "d:\Semester1\Semester 1 APro\Go\MODUL13\Latsol1.go"
Masukkan bilangan bulat positif: 5
1
PS D:\Semester1\Semester 1 APro\Go\MODUL13> go run "d:\Semester1\Semester 1 APro\Go\MODUL13\Latsol1.go"
Masukkan bilangan bulat positif: 234
3
PS D:\Semester1\Semester 1 APro\Go\MODUL13> go run "d:\Semester1\Semester 1 APro\Go\MODUL13\Latsol1.go"
Masukkan bilangan bulat positif: 78787
5
```

Deskripsi Program:

Program Latsol1.go dibuat untuk menghitung banyaknya digit dari suatu bilangan.

Masukan berupa bilangan bulat positif.

Keluaran berupa bilangan bulat yang menyatakan banyaknya digit dari bilangan yang diberikan pada masukan. User dapat menginputkan data yang diinginkan didalam bagian terminal setelah program dijalankan dengan runner program. Berikut input dan ouput;

No	Masukan	Keluaran
1	5	1
2	234	3
3	78787	5
4	1894256	7

2. Latihan Soal 2

Source Code:

```
package main

import (
    "fmt"
    "math"
)

func main() {

    var masukanDesimal float64

    fmt.Print("Masukkan bilangan desimal: ")
    fmt.Scan(&masukanDesimal)

    //membuat ke atas angka input
    //ceil -> untuk menentukan bilangan bulat ke atas dari input
    atas := math.Ceil(masukanDesimal)

    //inisialisasi nilai awal
    nilai := masukanDesimal

    //loop hingga mencapai bilangan bulat (1)
    for nilai < atas {
        nilai = math.Round((nilai+0.1)*10) / 10
        fmt.Printf("%.1f\n", nilai)
    }

}
```

Output:

```
PS D:\Semester1\Semester 1 AIPro\Go\MODUL13> go run "d:\Semester1\Semester 1 AIPro\Go\MODUL13\Latsol2.go"
Masukkan bilangan desimal: 0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
PS D:\Semester1\Semester 1 AIPro\Go\MODUL13> go run "d:\Semester1\Semester 1 AIPro\Go\MODUL13\Latsol2.go"
Masukkan bilangan desimal: 2.7
2.8
2.9
3.0
```

Deskripsi Program:

Program Latsol2.go dibuat untuk mendapatkan bilangan bulat optimal dari bilangan yang telah diinputkan. Melakukan penjumlahan tiap perulangannya mencapai pembulatan keatas dari bilangan yang diinputkan

Masukan berupa bilangan desimal.

Keluaran terdiri dari bilangan hasil penjumlahan tiap perulangannya sampai pembulatan keatas dari bilangan yang diinputkan. User dapat menginputkan data yang diinginkan didalam bagian terminal setelah program dijalankan dengan runner program. Berikut input dan ouput;

No	Masukan	Keluaran
1	0.2	0.3
		0.4
		0.5
		0.6
		0.7
		0.8
		0.9
		1
2	2.7	2.8
		2.9
		3

3. Latihan Soal 3

Source Code:

```
package main

import "fmt"

func main() {
    var target, donasi, totalDonasi, jumlahDonatur int
    fmt.Scan(&target)

    for totalDonasi < target {
        fmt.Scan(&donasi)
        jumlahDonatur++
        totalDonasi += donasi
        fmt.Printf("Donatur %d: Menyumbang %d. Total terkumpul: %d\n",
            jumlahDonatur, donasi, totalDonasi)
    }
    fmt.Printf("Target tercapai! Total donasi: %d dari %d donatur.\n", totalDonasi,
        jumlahDonatur)
}
```

Output:

```
PS D:\Semester1\Semester 1 APro\Go\MODUL13> go run "d:\Semester1\Semester 1 APro\Go\MODUL13\Latsol3.go"
300
100
Donatur 1: Menyumbang 100. Total terkumpul: 100
50
Donatur 2: Menyumbang 50. Total terkumpul: 150
200
Donatur 3: Menyumbang 200. Total terkumpul: 350
Target tercapai! Total donasi: 350 dari 3 donatur.
PS D:\Semester1\Semester 1 APro\Go\MODUL13> |
```

```
PS D:\Semester1\Semester 1 APro\Go\MODUL13> go run "d:\Semester1\Semester 1 APro\Go\MODUL13\Latsol3.go"
500
150
Donatur 1: Menyumbang 150. Total terkumpul: 150
100
Donatur 2: Menyumbang 100. Total terkumpul: 250
50
Donatur 3: Menyumbang 50. Total terkumpul: 300
300
Donatur 4: Menyumbang 300. Total terkumpul: 600
Target tercapai! Total donasi: 600 dari 4 donatur.
|
```

```
PS D:\Semester1\Semester 1 APro\Go\MODUL13> go run "d:\Semester1\Semester 1 APro\Go\MODUL13\Latsol3.go"
200
300
Donatur 1: Menyumbang 300. Total terkumpul: 300
Target tercapai! Total donasi: 300 dari 1 donatur.
PS D:\Semester1\Semester 1 APro\Go\MODUL13> |
```

Deskripsi Program:

Program Latsol3.go untuk meminta input dari pengguna untuk jumlah donasi hingga total donasi mencapai atau melebihi target yang telah ditentukan..

Masukan pada baris pertama berupa bilangan bulat yang merupakan target donasi yang harus dicapai. Masukan pada baris berikut dan seterusnya merupakan bilangan bulat yang menyatakan donasi oleh setiap donatur, masukan terus diterima hingga target tercapai.

Keluaran berupa bilangan hasil total penjumlahan tiap perulangannya serta jumlah donatur.

User dapat menginputkan data yang diinginkan didalam bagian terminal setelah program dijalankan dengan runner program. Berikut input dan ouput;

No	Masukan	Keluaran
1	300 100 50 200	Donatur 1: Menyumbang 100. Total terkumpul: 100 Donatur 2: Menyumbang 50. Total terkumpul: 150 Donatur 3: Menyumbang 200. Total terkumpul: 350 Target tercapai! Total donasi: 350 dari 3 donatur.
2	500 150 100 50 300	Donatur 1: Menyumbang 150. Total terkumpul: 150 Donatur 2: Menyumbang 100. Total terkumpul: 250 Donatur 3: Menyumbang 50. Total terkumpul: 300 Donatur 4: Menyumbang 300. Total terkumpul: 600 Target tercapai! Total donasi: 600 dari 4 donatur.
3	200 300	Donatur 1: Menyumbang 300. Total terkumpul: 300 Target tercapai! Total donasi: 300 dari 1 donatur.

DAFTAR PUSTAKA

<file:///D:/Semester%201%20AlPro/Modul/MODUL%202.pdf>

<https://www.toppr.com/guides/computer-science/introduction-to-c/data-types-variables-and-constants/declaration-of-variables/>

<https://www.toppr.com/guides/computer-science/introduction-to-c/data-types-variables-and-constants/constants-in-programming-language/>

<https://www.dicoding.com/blog/macam-macam-tipe-data/>

<file:///D:/Semester%201%20AlPro/Modul/MODUL%203.pdf>

<file:///D:/Semester1/Semester%201%20AlPro/Modul/MODUL%205%20dan%206.pdf>

<file:///D:/Semester1/Semester%201%20AlPro/WEEKMATERI/MODUL%209.pdf>

<file:///D:/Semester1/Semester%201%20AlPro/WEEKMATERI/MODUL%2011.pdf>

<file:///C:/Users/Laptopku/Downloads/MODUL%2012.pdf>

<file:///C:/Users/Laptopku/Downloads/MODUL%2013.pdf>