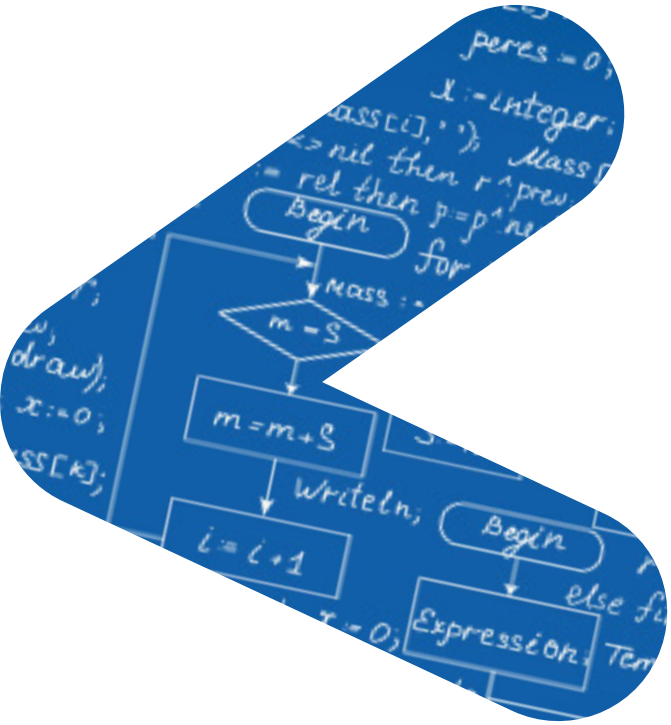
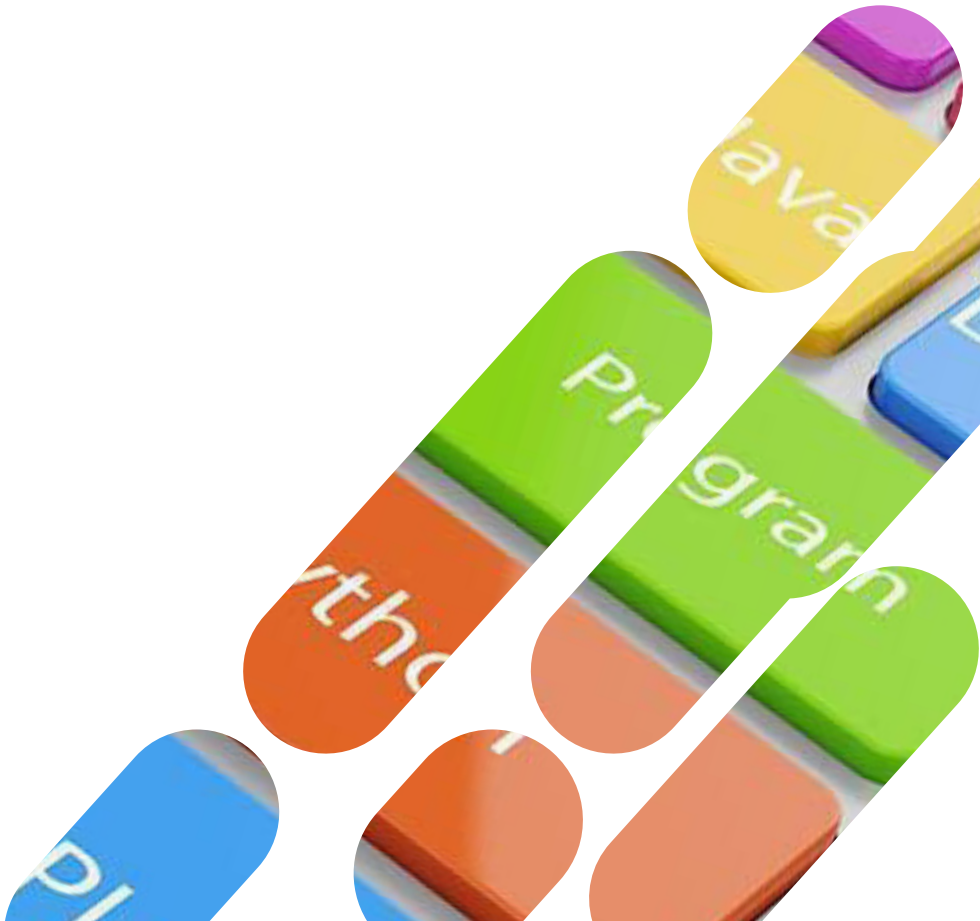


SMK Telkom Makassar

PERULANGAN

PEMROGRAMAN TERSTRUKTUR



PEMROGRAMAN TERSTRUKTUR

PERULANGAN

Rekaya Perangkat Lunak SMK Telkom Makas*sar*

Tujuan Pembelajaran

1. Peserta didik dapat Memahami struktur perulangan
2. Perserta didik dapat Membuat Program Sederhana Dengan perulangan

Perulangan adalah salah satu konsep yang paling kuat dalam dunia pemrograman. Perulangan adalah cara Anda memberi tahu komputer untuk mengulangi serangkaian operasi sebanyak yang diperlukan.

Mengapa Perulangan Digunakan?

Perulangan digunakan ketika berapa kali operasi harus dilakukan tidak diketahui, tetapi Anda tahu kapan harus berhenti.

Manusia juga berpikir dalam lingkaran - Anda tidak menyadarinya! Berhentilah sejenak untuk berpikir tentang bagaimana Anda menghitung uang. Bayangkan Anda membuka dompet atau tas Anda. Sekarang, bayangkan ada uang fisik di dalamnya. Bagaimana Anda akan menghitung jumlah total dolarnya?

while

Perulangan while adalah salah satu jenis perulangan yang ada dalam JavaScript.

```
while (condition) {
 }
```

Perulangan while akan terus berulang sampai kondisi bernilai false.

STOPPING A LOOP

Kondisi dalam perulangan juga disebut sebagai kondisi penghentian perulangan. Perulangan dihentikan dengan sengaja mengarahkannya ke nilai yang salah. Setelah nilainya salah, perulangan akan berakhir dan sisa kode di bawah perulangan akan terus beroperasi.

Loop Logic

Aturan menghitung uang dapat membantu Anda dengan logika. Namun sekarang Anda perlu mengetahui operasi yang sebenarnya dapat dilakukan oleh komputer.

Pertama, siapkan sebuah dompet yang penuh dengan uang kertas. Tetapi konsep dompet tidak ada dalam JavaScript. Sebagai gantinya, Anda akan memiliki serangkaian uang kertas.

Setelah memeriksa dompet Anda, Anda akan melihat dua lembar uang kertas $1, satu lembar uang kertas $5, satu lembar uang kertas $10, dan satu lembar uang kertas $20. Jadi, susunan uang kertas Anda akan terlihat seperti ini.

while

```
var bills = [1, 1, 5, 10, 20];
```

Di sinilah letaknya sedikit rumit. Sebagai programmer, kita perlu menggunakan aturan komputer. Hal ini membutuhkan sedikit kreativitas. Beginilah cara Anda memodifikasinya:

1. Membuat larik tagihan.
2. Mengakses nilai dari setiap item dalam array, berdasarkan indeksnya.

1. Buat variabel yang akan mengakumulasikan nilainya dari setiap item dalam larik.
2. Membuat variabel yang bertujuan untuk mendapatkan setiap item dalam larik berdasarkan indeksnya. Penambahan setelah setiap tagihan.
3. Selagi variabel indeks masih berada di dalam batas-batas larik, terus lakukan pengulangan.

Iteration

Sebelum menghitung, tulis satu putaran yang mencetak setiap tagihan. Juga memvalidasi indeks.

Prepare the array of bills and declaration of the index, which starts at 0.

```
// array of bills
var bills = [1, 1, 5, 10, 20];
// index to access array values
var index = 0;
```

Property

Ingat diskusi sebelumnya tentang properti panjang pada array? Ini adalah kunci dari semua keajaiban. Output angka dari panjang akan berubah berdasarkan ukuran larik. Hal ini akan memberitahukan kepada Anda seberapa besar larik tersebut tanpa Anda perlu melakukan pekerjaan tambahan!

```
bills.length; // 5
```

Saat perulangan beroperasi, indeks dan panjang larik dibandingkan. Ketika nilai indeks kurang dari panjang larik, berarti perulangan masih dalam batas larik.

```
// our terminating condition
 index < bills.length;
```

While index < bills.length terus bernilai benar, perulangan harus dilanjutkan.

```
// array of bills
var bills = [1, 1, 5, 10, 20];
// index to access array values
var index = 0;

// terminating condition
while (index < bills.length) {
// increment index to eventually meet
our terminating condition
index++;
}
```

Anda sekarang memiliki perulangan kode yang akan diakhiri. Tetapi tidak ada hal menarik yang terjadi di dalamnya. Sebelum menyelesaikan masalah awal Anda, pastikan semua berjalan dengan pernyataan console.log.

```
// array of bills
var bills = [1, 1, 5, 10, 20];
// index to access array values
var index = 0;
// terminating condition
while (index < bills.length) {
     var currentBill = bills[index];
     var indexPrint = 'I am on index' + index;
     console.log(indexPrint);
     var printedBill = 'I am currently on a $' +
   currentBill + 'bill!';
     console.log(printedBill);
// increment index to eventually meet our terminat-
ing condition

index++;
}
```

for

Perulangan while adalah titik masuk yang bagus ke dalam perulangan, tetapi perulangan for lebih umum digunakan. Ini mencakup situasi yang paling umum dengan sintaks yang ringkas.

STRUCTURE
Sintaksnya adalah sebagai berikut. Perhatikan titik koma.

```
for (initialization; condition;after) {

}
```

1. Inisialisasi: Operasi yang dilakukan sebelum iterasi apa pun. Ini hanya dilakukan sekali.
2. Kondisi: Kondisi pengakhiran yang diperiksa tepat sebelum iterasi berikutnya.
3. Setelah: Operasi yang dilakukan setelah iterasi. Ini dilakukan setelah iterasi peristiwa.

WHILE LOOP VS. FOR LOOP SYNTAX
Mari kita tinjau kembali strategi untuk mengulang sebuah larik dalam perulangan perulangan sementara, dalam pseu code

```
inisialisasi indeks
 while (indeks kurang dari
 array.length) {
   indeks kenaikan
}
```

1. Inisialisasi variabel untuk melacak indeks
2. Mengakhiri kondisi
3. Variabel indeks kenaikan

Perulangan for mengambil operasi-operasi ini dan memadatkannya ke dalam satu baris. Setiap operasi dipisahkan dengan titik koma.

```
for (inisialisasi index; index is
 kurang dari array.length;
 indeks kenaikan) {
}
```

Contoh perulangan while sebelumnya dapat dapat ditulis sebagai perulangan for.

```
var bills = [1, 1, 5, 10, 20];
var index = 0;

while (index < bills.length) {
   console.log(bills[index]);
   index++;
}
```

Semuanya sama, hanya saja lebih pendek

```
var bills = [1, 1, 5, 10, 20];
for (var index = 0; index <
 bills.length; index++) {
   console.log(bills[index]);
}
```

Loop Modifying Keywords

Ada beberapa algoritme yang masuk akal untuk melewatkan operasi berikutnya dalam sebuah perulangan dan pindah ke perulangan berikutnya atau langsung meninggalkan perulangan tersebut.

CONTINUE

Setelah pernyataan lanjutkan ditemukan, iterasi saat ini akan berhenti dan berlanjut ke iterasi berikutnya. Mari kita ubah prompt kita agar kita dapat menggunakan continue dalam penyelesaiannya.

Hanya mencetak uang kertas dengan nilai $5 atau lebih besar. Bagaimana hal itu dinyatakan dalam kode?

```
var bills = [1, 1, 5, 10, 20]; for (var index = 0;
index <
 bills.length; index++) {
   if (bills[index] >= 5) {
   console.log(bills[index]);
   }
}
```

Hal yang sederhana. Tetapi Anda dapat menggunakan pernyataan continue untuk menyelesaikan tantangan yang sama. Seperti inilah tampilan dari pseudo kode.

```
for (perulangan pada setiap tagihan) {
   jika tagihan kurang dari 5
   dolar, lewati semuanya
   di bawah pernyataan ini
   cetak tagihan
}
```

Jika iterasi berhenti ketika tagihan kurang dari $5, berarti tagihan tidak akan dicetak. Dalam kode yang sebenarnya, akan terlihat seperti ini.

```
var bills = [1, 1, 5, 10, 20]; for (var index = 0;
index <
bills.length; index++) {
// if the bill is less than 5 dollars if (bills[in-
dex] < 5) {
// skip
continue; }
   console.log(bills[index]);
 }
```

BREAK

Kata kunci continue akan melewatkan operasi selanjutnya di dalam perulangan. Tetapi bagaimana jika Anda ingin membatalkan perulangan sepenuhnya? Pernyataan break menghentikan perulangan saat ini dengan cara yang sama seperti continue. Tetapi break melangkah lebih jauh lagi dan menghentikan semua perulangan berikutnya juga!

Sekarang, ubah prompt sekali lagi. Jika Anda melihat uang kertas $5, hentikan pencetakan. Hal ini dapat dilakukan tanpa pernyataan jeda. Tetapi dengan menggunakan pernyataan jeda akan membuat algoritme lebih mudah dipahami.

```
for (perulangan pada setiap tagihan) {
   jika tagihan sama dengan 5
dolar, keluar dari perulangan mencetak tagihan
```

Dalam kode yang sebenarnya, akan terlihat seperti ini.

```
var bills = [1, 1, 5, 10, 20]; for (var index = 0;
index < bills.length; index++) {
// if the bill is 5 dollars
if (bills[index] === 5) {
// exit loop completely
break; }
   console.log(bills[index]);
 }
```

action

```
const input=require('readline-sync')
var nama,umur
do{
    nama=input.question('masukan nama anda:')
    umur=input.questionInt('umur:')

console.log(`Halo ${nama} anda berumur ${umur}`)
var inputLagi=input.keyInYNStrict(`input data
lagi`)

}while(inputLagi)

console.log('selesai')
```