05 Perulangan

Ringkasan Materi

Struktur perulangan (*loop*, iterasi) dalam pembelajaran algoritma adalah mekanisme yang memungkinkan suatu blok kode dieksekusi berulang kali selama kondisi tertentu terpenuhi. Perulangan membantu menghindari penulisan kode yang berulang secara manual dan sangat berguna untuk memproses data berurutan (seperti daftar atau array), menggambar pola berulang, atau menjalankan simulasi.

Struktur perulangan memiliki beberapa bentuk di mana masing – masing memiliki cara kerja yang sesuai untuk skenario berbeda. Pemahaman perulangan merupakan fondasi penting dalam membangun program yang efisien dan dinamis.

Komponen utama dalam sebuah perulangan meliputi:

1. Inisialisasi, menetapkan nilai awal
2. Kondisi, berupa nilai logika (*true / false*) untuk menentukan apakah perulangan berlanjut atau berhenti.
3. Langkah, berupa *update* nilai pengontrol untuk menentukan kapan perlangan akan berhentu.

Beberapa bentuk perulangan adalah

1. for
   Struktur **for** akan mengeksekusi statement dengan jumlah iterasi yang diketahui, di mulai dari nilai inisial dengan langkah tertentu.
2. while
   Struktur **while** hampir sama dengan for tetapi memiliki bentuk formal diperuntukkan bagi iterasi yang jumlahnya tidak pasti.
3. do – while
   Struktur **do – while** hampir sama dengan while, tetapi kondisi diperiksa di akhir.

Penjelasan selengkapnya dapat dilihat pada slide **05 Perulangan.pptx**.

Percobaan

AsciiShow

Menampilkan 256 karakter ASCII (American Standard Code for Information Interchange) menggunakan perulangan FOR. Pada dasarnya semua karakter, baik ada di keyboard atau tidak, yang dapat ditampilkan melalui konsol memiliki kode ASCII. Sebagai contoh adalah huruf A memiliki kode ASCII 65.

Karakter ASCII ini nantinya dapat digunakan untuk membuat tampilan di konsol lebih menarik, seperti membuat *layout* tabel, emoji, dan efek – efek lainnya. Contoh penggunaannya adalah sebagai berikut ini:

```
cout << char(65);
```

Kode di atas akan menuliskan huruf A (kode ASCII 65) di konsol.

Berikut ini adalah potongan lengkap kode untuk menampilkan keseluruhan kode ASCII:

```
for(int i = 0; i < 256; ++i)
{
    cout << “Kode: “ << i << ‘\t’;
    cout << char(i) << endl;
}
```

Pyramid

Demo menampilkan bentuk piramida atau segi tiga menggunakan struktur perulangan bersarang. Piramida disusun dari karakter ‘*’ dari atas ke bawah. Tinggi piramida berdasar input dari *user*. Berikut ini adalah tampilan outputnya:

```
height: 7
*
**
***
****
*****
******
*******
```

Potongan kode selengkapnya adalah sebagai berikut:

```
int height;
Cout >> “height: “;
cin >> height;

for(int row = 1; row <= height; ++row)
{
    for(int col = 1; col <= row; ++col)
    {
        Cout << ‘*’;
    }
    cout << endl;
}
```

Untuk menampilkan piramida dengan bentuk yang lain seperti berikut ini

```
Height: 7
       *
      ***
     *****
    *******
   *********
  ***********
*************
```

Dapat dilakukan dengan memodifikasi di bagian perulangan. Potongan kode selengkapnya adalah sebagai berikut:

```
int height;

cout << "height: ";
cin >> height;
```

```
for (int row = 1; row <= height; ++row) {
          // Print leading spaces
          for (int col = 0; col < height - row; ++col) {
                    cout << " ";
          }
          // Print stars
          for (int col = 0; col < (2 * row - 1); ++col) {
                    cout << "*";
          }
          cout << endl;
}
```

Ubahlah karakter '*' menggunakan karakter **ASCII 219.**

RepLetter

Project ini akan mengubah semua huruf vokal dari sebuah kalimat yang diinput. *User* diminta untuk memberi input berupa kalimat kemudian outputnya berupa kalimat yang huruf vokalnya sudah diganti dengan karakter '*'.

Contoh tampilannya adalah sebagai berikut:

```
Input : universitas amikom yogyakarta
Output: *n*v*rs*t*s *m*k*m y*gy*k*rt*
```

Tipe data string merupakan rangkaian dari karakter (tipe char). Tiap karakter yang ada dalam sebuah string memiliki nomor urut (indeks) yang di mulai dari 0. Tipe string memiliki sebuah fungsi **length()** untuk meghitung panjang karakter yang ada di dalamnya. Sebagai contoh string "amikom" memiliki *length* 6 karakter, dengan indeks dimulai dari 0 hingga 5.

Indeks	0	1	2	3	4	5
karakter	a	m	i	k	o	m

Sebagai contoh, bila string "amikom" disimpan pada variabel dengan nama **myStr**, untuk mengakses huruf **'k'** dengan indeks 3 bisa menggunakan format **myStr[3].**

Berikut ini adalah potongan kode selengkapnya:

```
string sentence;
const string vowels = "aeiouAEIOU";

cout << "Enter a sentence : ";
getline(cin, sentence);

for(int i = 0; i < sentence.length(); ++i)
{
    for(int j = 0; j < vowels.length(); ++j)
    {
        if(sentence[i] == vowels[j])
        {
            sentence[i] = '*';
            break;
        }
    }
}

cout << "Modified sentence: " << sentence << endl;
```

```
system("pause");
```

Penjelasan:

Terdapat variabel **sentence** untuk menyimpan input dan konstanta **vowels** untuk daftar huruf vokal. Tiap huruf di variabel **sentence** (menggunakan perulangan luar, i) akan dicocokkan satu per satu dengan semua huruf di **vowels (**menggunakan perulangan dalam, j). Bila ada yang cocok, maka huruf pada iterasi yang **ke-i** akan diganti dengan '*'.

StopWatch

Membuat **stopwatch** digital memanfaatkan fungsi **Sleep().** Fungsi ini akan memperlambat (*delay*) sebuah proses dalam satuan *millisecond*. Dalam prosesnya diperlukan variabel yang mencatat siklus perulangan tiap 1 detik (1000 *milliseconds*). Nilai variabel akan selalu naik 1 tiap 1 detik. Berikut ini adalah potongan kode selengkapnya:

```
int seconds = 0;
int minutes = 0;
int hours = 0;

while (true)
{
    system("cls");
    // display the time in HH:MM:SS format
    cout << setfill('0') << setw(2) << hours << ":"
         << setfill('0') << setw(2) << minutes << ":"
         << setfill('0') << setw(2) << seconds << endl;

    Sleep(1000);
    seconds++;
    // if seconds reach 60, increment minutes and reset seconds
    if(seconds >= 60)
    {
        seconds = 0;
        minutes++;
    }
    // if minutes reach 60, increment hours and reset minutes
    if(minutes >= 60)
    {
       minutes = 0;
       hours++;
    }
    // if hours reach 24, reset hours
    if(hours >= 24)
    {
        hours = 0;
    }
}
```

Kode di atas dapat dikembangkan untuk ditambah aksi **start, stop,** dan **reset**. Aksi ini dapat dilakukan melalui input melalui penekanan sebuah tombol (karakter) di *keyboard*. Penekanan tombol ini tidak memerlukan ENTER. Sebagai contoh, untuk aksi **start** menggunakan tombol 'a'. Ketika *user* menekan tombol 'a' (tanpa ENTER), *stopwatch* akan langsung mulai. Fungsi yang digunakan utuk pembacaan input tanpa ENTER adalah **_getch()** dari *header* **<conio.h>**. Contoh penggunaannya adalah:

```
char key = _getch();
```

Mulailah bereksplorasi...

Latihan

1. Buatlah sebuah *project* untuk memeriksa apakah sebuah bilangan yang diinput termasuk prima atau bukan. Contoh tampilannya adalah sebagai berikut:

```
Input: 49
49 is not prime number

Input: 47
47 is prime number
```

2. Buatlah sebuah *project* untuk memeriksa apakah sebuah string yang diinput termasuk palindrom atau bukan. Contoh tampilannya adalah sebagai berikut:

```
Input: kodok
“kodok” is  palindrome

Input: tomat
“tomat” is not palindrome
```

Catatan:
Palindrom adalah sebuah teks dibaca dari depan atau depan berbunyi sama.