

[Home](#) / [Tutorial](#) / [C++](#) / [Pernyataan Goto](#)

Pernyataan Goto

Posted on Januari 23, 2018 | Last Modified Januari 23, 2018



Pernyataan GOTO adalah pernyataan yang memungkinkan kita untuk mengatur arahnya aliran pengesekusian CPU terhadap program kita. GOTO berfungsi untuk memerintahkan CPU melompat ke baris manapun berdasarkan label yang telah dibuat.

Bentuk Penulisan Goto

```
goto nama_label;
```

Bentuk penulisan Label Goto

```
nama_label:
```

untuk membuat pernyataan goto bekerja, dibutuhkan dua pernyataan yaitu GOTO dan LABEL. Pernyataan GOTO menggunakan keyword "goto" diikuti nama label berfungsi untuk memberi tahu CPU agar melompat ke baris yang memiliki label dengan nama tersebut.

Pernyataan LABEL diisi dengan nama label diakhiri dengan tanda titik dua, berfungsi untuk menandai suatu baris program, tempat dimana loncatan CPU karena GOTO mendarat.

Contoh Penulisan Goto

```
goto labelku;
```

Contoh Penulisan Label Goto

```
labelku:
```

Ketika CPU bertemu dengan pernyataan GOTO maka CPU pada saat itu juga akan melompat ke label dengan nama yang tertera pada pernyataan GOTO. Dan untuk penempatan GOTO beserta label GOTO kita bebas meletaknya dimana saja tetapi harus berada pada satu function scope.

Contoh Program :

```
#include <iostream>
using namespace std;

int main ()
{
    int angka=0;
    cobaLagi:
    cout<<"Masukan Angka : ";cin>>angka;
    if (angka!=5)goto cobaLagi;

    return 0;
}
```

Contoh program di atas adalah contoh penggunaan pernyataan GOTO. Program di atas akan meminta pengguna untuk memasukkan angka apapun, jika pengguna memasukkan angka 5 maka pernyataan GOTO akan dieksekusi, CPU akan melompat ke label "cobaLagi" yang diletakan di atas.

Peletakan Label dari GOTO bisa diletakan di baris sebelum pernyataan GOTO, kita juga bisa meletakan Label GOTO di baris setelah dari pernyataan GOTO.

Contoh Program :

```
#include <iostream>
using namespace std;

int main ()
{
    char loncat=0;

    cout<<"Masukan Angka [y/n] : ";cin>>loncat;
    cout <<"1"<<endl;
    if (loncat=='y')goto loncatan;
    cout <<"2"<<endl;
    loncatan:
    cout <<"3"<<endl;

    return 0;
}
```

Contoh program di atas akan meminta persetujuan anda untuk melompat, jika anda menginputkan 'n' maka aliran akan berjalan normal dan menghitung 1 sampai 3, jika anda memilih y maka akan tampil perhitungan 1 dan 3.

Setelah penjelasan mengenai GOTO dan cara penggunaan GOTO, penulis belajarcpp.com sarankan untuk tidak menggunakan pernyataan GOTO jika tidak dibutuhkan. Sebenarnya bukan hanya penulis yang menyarankan, tapi hamper semua programmer menghindari penggunaan pernyataan GOTO.

Pernyataan GOTO dapat menurunkan kualitas program, karena bisa membuat program mempunyai aliran yang sangat tidak jelas. Kadang hal itu bisa menyebabkan kegagalan pada program anda.

Di bahasa pemrograman lainnya, salah satunya seperti bahasa pemrograman BASIC, Goto sangat sering digunakan, hal itu normal di bahasa pemrograman BASIC tapi berbeda pada Bahasa pemrograman C++, semua programmer yang menggunakan bahasa pemrograman C++ sama sekali tidak menggunakan pernyataan GOTO, mereka mengganti pernyataan GOTO dengan memanfaatkan pernyataan lain yang lebih canggih dan aman seperti pernyataan pengulangan, penyeleksian dan lain-lain.


Dan di bawah ini adalah salah satu contoh program mengapa GOTO tidak disarankan untuk digunakan.

```
#include <iostream>
using namespace std;


int main ()
{
    goto loncat;
    int var = 2;
    loncat:
    cout<<var;
    return 0;
}
```


Program di atas tidak akan bisa dijalankan karena CPU akan dilempar langsung ke baris pada label loncat dengan arti deklarasi var tidak pernah di eksekusi, dan CPU tidak bisa menemukan variabel var saat pernyataan keluaran.

Bayangkan jika anda membuat program dengan kode yang cukup banyak, dan anda menggunakan begitu pernyataan GOTO. Kemungkinan yang akan terjadi adalah anda akan bingung bagaimana CPU akan mengalir untuk mengeksekusi program anda, anda sulit untuk menyunting program anda dan kemungkinan program akan mendapatkan kesalahan yang cukup fatal karena aliran program yang tidak jelas.

 Share

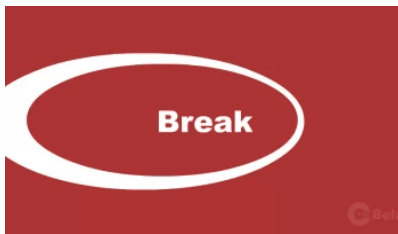
 Tweet

 Share

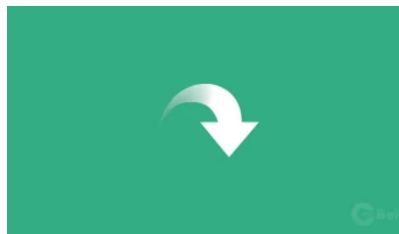
 Pin

 Share

Terkait



Pernyataan Break
Januari 23, 2018
dalam "C++"



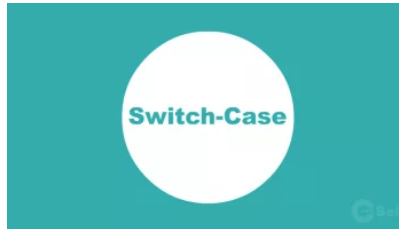
Pernyataan Lompatan
Januari 23, 2018
dalam "C++"



Pernyataan dan Kontrol Aliran
Januari 23, 2018
dalam "C++"



Pernyataan Continue
Januari 23, 2018
dalam "C++"



Pernyataan Switch
Januari 23, 2018
dalam "C++"



Pernyataan Do-while
Januari 23, 2018
dalam "C++"