**MAKALAH PERULANGAN**



DISUSUN OLEH :

Nama : L Hafidl Alkhair

NIM : 2023903430060

Kelas : TRKJ 1.C

Jurusan : Teknologi Informasi dan Komputer

Program Studi : Teknologi Rekayasa Komputer Jaringan

Dosen Pengajar : Indrawati, SST. MT



***PROGRAM STUDI TEKNOLOGI REKAYASA KOMPUTER JARINGAN***

***JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER***

***POLITEKNIK NEGERI LHOKSEUMAWE***

***TAHUN 2022/2023***

**PENGULANGAN PROSES**

Mengulang suatu proses merupakan Tindakan yang banyak dijumpai dalam pemrograman. Pada semua Bahasa pemrograman, pengulangan proses ditangani dengan suatu mekanisme yang disebut loop. Dengan menggunakan loop, suatu proses yang berulang misalnya menampilkan tulisan yang sama serratus kali pada layer dapat diimplementasikan dengan kode program yang pendek

Pada C terdapat tiga buah struktur loop. Pernyataan yang dipakai berupa while, do while dan for. Berhubugan dengan penanganan loop ada beberapa pernyatan lain yang berfungsi diantaranya untuk keluar dari loop adalah break, goto dan continue.

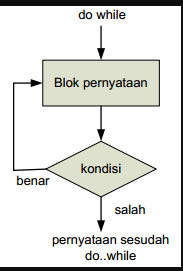
**PERNYATAAN DO-WHILE.**

Bentuk pernyataan do-while

**do**

**pernyataan**

**while(kondisi)**

pada pernyataan do-while, tubuh loop berupa pernyataan, dengan pernyataan bisa berupa pernyataan Tunggal, pernyataan meajemuk atapun pernyataan kosong. Pada pernyataan do, mula-mula pernyataan dijalankan. Selanjutnya, kondisi diuji. Seandainya kondisi bernilai benar, maka pernyataan dijalan kan lagi, kemudian kondisi di periksa kembali, dan seterusnya. Kalau kondisi salah pada saat dites, maka pernyataan tidak dijalankan lagi.

Program berikut ini memberikan contoh pemakain do-while untuk mengatur penampilan tulisan” HAFIDL PASTI BISA” sebanyak sepuluh kali

**#include <stdio.h>**

**main(){**

**int N;**

**N =0;**

**do**

**{**

**puts("HAFIDL PASTI BISA ");**

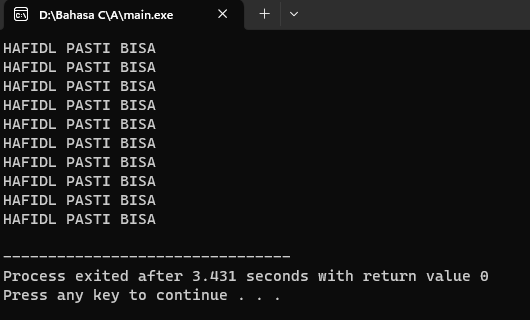
**N++;**

**}**

**while ( N < 10);**

**}**

OUTPUT



Program tersebut, Variable N dipakai untuk menghitung jumlah tulisan yang sudah ditampilkan pada layar. Selama nilai N kurang dari 10 , maka perintah

**puts("HAFIDL PASTI BISA ");**

akan di laksanakan kembali.

Contoh lain pemakain do-while misalnya untuk mengatur tombol yang di tekan oleh pemakai program berupa salah satu diantara ‘Y’ , ‘ y’, ‘ T’ atau ‘t’. implementasinya:

**do {**

**P = getch();**

**Sudah\_benar = (P == 'Y') || ( P == 'y') ||**

**P == 'T') || ( P == 't');**

**}**

**while(!Sudah\_benar);**

Mula-mula tombol dibaca dengan menggunakan getch() dan kemudian diberikan ke variable P. Sesudah itu, variable sudah\_benar akan diisi dengan nilai benar (1) atau salah ( 0 ) tergantung dari nilai pilihan. Kalau P berisi salah satu diantar ‘Y’ ‘y’, ‘t’ maka sudah\_benar akan berisi benar. Nilai pada variable sudah\_benar ini selanjutnya dijadikan sebagai kondisi do-while. Pengulangan terhadap pembacaan tombol akan dilakukan kembali selama sudah\_benar bernilai salah.

#include <stdio.h>

#include <conio.h>

main(){

char P;

int B;

printf("Chose Y or T. ");

// program hanya mau dilanjutkan jika button y,Y,t yang di tekan

do {

P = getch(); // baca tombol

B = (P == 'Y') || ( P == 'y') ||

(P == 'T') || ( P == 't');

}

while(!B);

// ket about chose

switch(P){

case 'y':

case 'Y':

puts("\n Your chose is Y");

break;

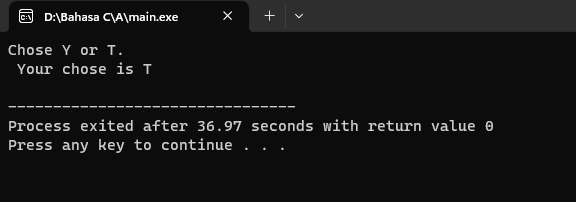
case 't':

case 'T':

puts("\n Your chose is T");

}

}



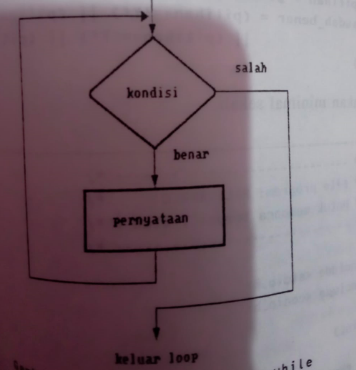
Program di atas terdapat tambahan “Swit Case” yang dimana bertujuan untuk variable P bisa memilih keterangan tentang pilihan

**PERNYATAAN WHILE**

Berbeda dengan do-while, pada pernyataan while pengujian terhadap loop dilakukan di bagian awal ( sebelum tubuh loop ) lebih jelasnya, bentuk pernyataan while adalah sebagai berikut:

While ( kondisi )

Penyataan

Dengan pernyataan dapat berupa pernyataan tunggal, pernyataan majemuk ataupun pernyataan kosong, proses pengulangan terhadap *penyataan*

Tampak bahwa ada kemungkinan *pernyataan*  yang merupakan tubuh loop tidak dijalan kan sama sekali, yaitu kalau pada pengujian kondisi **while** yang pertama kali ternyata kondisi bernilai salah.

Penanganan pembacaan tombol pada program di atas yang memakai **do-while** kalau dilaksanakan dengan memakai **while** adalah sebagai berikut:

sudah\_benar = 0 ;

while (!sudah\_benar){

P = getch();

sudah\_benar = ( P == 'Y') || ( P == 'y')

||( P == 'T') || ( P == 't');

}

Perbedaan yang utama, di awali pernyataan **while** harus diberikan terlebih dahulu inisialisasi terhadap variable **sudah\_benar** yang akan dijadikan sebagai kondisi dari **while.** Variable **sudah benar** diberi nilai awal salah ( 0 ) dimaksudkan agar tubuh **loop**

**{**

**P = getch();**

**sudah\_benar = ( P == 'Y') || ( P == 'y') ||**

**( P == 'T') || ( P == 't');**

**}**

Dijalankan minimal sekali

#include <stdio.h>

#include <conio.h>

main(){

char pilihan;

int sudah\_benar;

printf("Pilih Y atau T. ");

sudah\_benar = 0;

while(!sudah\_benar)

{

pilihan= getch();

sudah\_benar = ( pilihan == 'Y') || ( pilihan == 'y')||

( pilihan == 'T') || ( pilihan == 't');

}

switch(pilihan){

case 'y':

case 'Y':

puts("\n Your chose is Y");

break;

case 't':

case 'T':

puts("\n Your chose is T");

}

}

Contoh lain pemakain while dapat dilihat pada program di atas , program tersebut digunakan untuk menghitung banyaknya karakter dari kaliamt yang dimasukan melalui papan-ketik. Disamping itu juga menghitung banyak karakter spasi. Untuk mengakhiri pemasuk kalimat, tombol ENTER harus ditekan Karena itu tombol ENTER inilah dijadikan sebagai kondisi perhitungan jumlah spasi maupun karakter seluruhnya. Lengkapnya kondisi yang di pakai dalam while berupa :

( karakter = getch() ) ! = ENTER

Ungkapan di atas mempunyai arti :

* Bacalah sebuah karakter dan berikan variable karakter
* Kemudian bandingkan apakah karakter tersebut tidak sama dengan ENTER ( catatan : ENTER didefinisikan dalam #define).

Ungkapan menghasilkan nilai benar jika tombol yang ditekan bukan ENTER.

Pada program, kalau tombol yang di tekan bukan ENTER, maka :

Jumlah karakter dinaikan sebesar satu ( jumkar++; )

Kalau karakter berupa SPASI, maka jumlah spasi dinaikan sebesar satu, melalui pernyataan

If ( karakter == SPASI ) Jumspasi++;

**Program menghitung jumlah kata dan karakter dalam suatu kalimat**

#include <stdio.h>

#include <conio.h>

#define ENTER '\r'

#define SPASI ‘ ‘

main(){

char karakter;

int jumkar = 0;

int jumspasi = 0;

puts("Masukan sebuah kalimat dan akhiri dengan ENTER");

puts("Saya menghitung jumlah karakter");

puts("Pada kalimat tersebut. \n");

while( (karakter = getche()) != ENTER )

{

jumkar++;

if (karakter == SPASI ) jumspasi++;

}

printf("\n Jumlah karakter = %d", jumkar);

printf("\n Jumlah SPASI = 2%d", jumspasi);

}

Program diatas jika di jalankan akan terjadi eror , dan jika kita hapus di bagian if (karakter == SPASI ) jumspasi++; maka bisa dijalan kan

**PERNYATAAN FOR.**

Pernyataan ketiga untuk keperluan pengulangan proses adalah pernyataan for betnuk pernyataan ini

For ( ungakapan1, ungakapan2, ungakapan3, )

pernyataan

kegunaan dari masing masing ungkapan pada pernyataan for.

1. ungakapan1 digunakan untuk memberikan inisialisia terhadap variable pengendali loop
2. ungakapan2 digunakan sebagai kondisi untuk keluar dari loop
3. ungakapan3 dipakai sebagai pengatur kenaikan nilai variable pengendali loop

ketiga ungkapan dalam **for** tersebut harus dipisahkan dengan tanda titik koma.

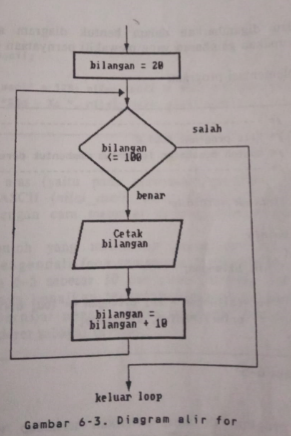
**Loop for** sebenarnya identik dengan bentuk **while** sebagai berikut:

**ungkapan1 ;**

**while ( ungkapan2){**

**pernyataan**

**ungkapan 3;**

**}**

Contoh penggunaan **for** misalnya untuk menampilkan deretan angka sebagai berikut:

20

30

40

50

60

:

100

Untuk keperluan ini, pernyataan **for** yang digunakan berupa

for ( bil = 20 ; bil <= 100 ; bil += 10 )

printf(" %d \n ", bil );

kalau digambarakan dalam bentuk diagram alir diatas merupakan gambaran yang mewakili pernyataan didepan

impelementasi program

#include <stdio.h>

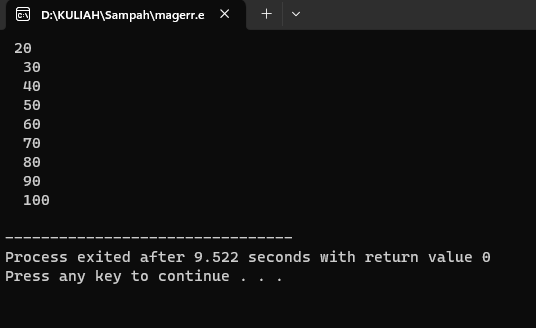
main(){

int bil;

for( bil =20 ; bil <= 100 ; bil += 10 )

printf(" %d\n ", bil );

}



pada program diatas, kenaikan terhadap variable pengendali **loop** sebesar 10

( positif )

Bil += 10 ( *atau* bil = bil + 10)

Contoh lain pemakain pernyataan **for** yaitu untuk menampilkan nilai ASCII

dan karakternya, sebagaimana ditunjukan pada program berikut :

**#include <stdio.h>**

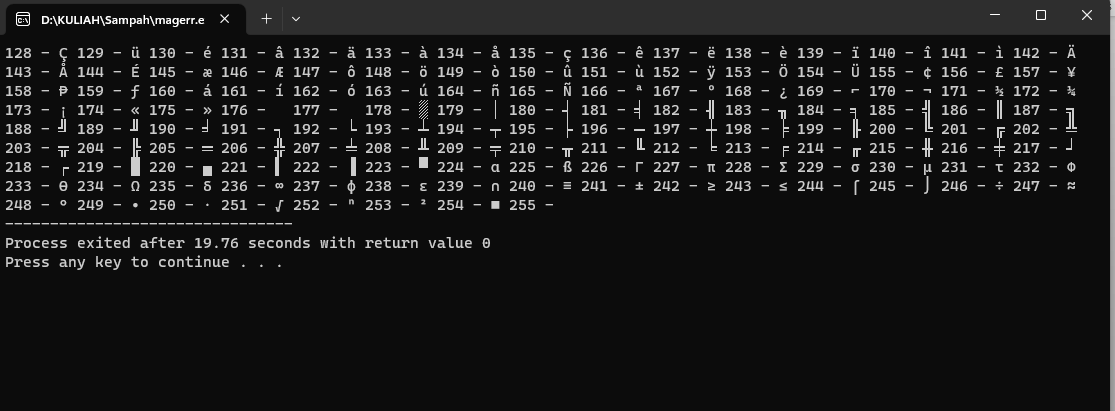
**main(){**

**int nilai\_ascii;**

**for( nilai\_ascii = 128 ; nilai\_ascii <= 255 ; ++nilai\_ascii )**

**printf("%3d - %c ", nilai\_ascii , nilai\_ascii);**

**}**

****

pada program diatas ( yaitu pada pernyataan printf()) terlibat bahwa nilai ASCII ( **nilai\_ascii**)dapat ditampilkan dalam bentuk karakter dengan cara memakai penentu format berupa **%c**

pada dua contoh melibatkan pernyataan **for** kenaikan variable pengendali loop semua berupa nilai positif.( pada program pertama sebesar 10 dan pada program kedua sebesar 1 )**.** Sebenarnya kenaikan pada variable pengendali LOOP bisa diatur bernilai negative . cara ini dapat digunakan untuk memperoleh deret sebagai berikut :

60

50

40

30

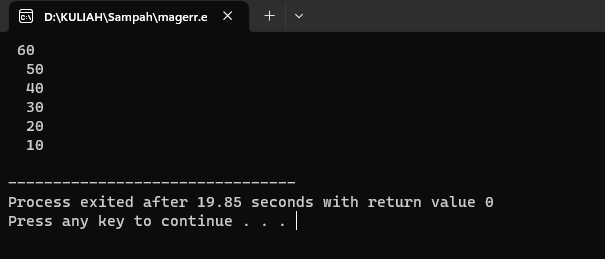
20

10

Pernyataan FOR yang diperlukan

for ( bil = 60 ; bil >= 10 ; bil -=10)

printf(" %d \n ", bil );



**LOOP DALAM LOOP**

Loop dalam loop, juga dikenal sebagai "nested loop," merujuk pada penggunaan satu atau lebih loop di dalam loop lainnya. Ini digunakan ketika Anda perlu melakukan perulangan dalam konteks yang lebih kompleks, seperti mengakses elemen dalam array multi-dimensi atau membuat pola tertentu.

Dalam bahasa C, ada beberapa jenis loop yang umum digunakan:

* Loop while: Loop ini akan terus menjalankan pernyataan yang ada dalam blok kode selama kondisi yang diberikan adalah benar. Loop ini cocok digunakan ketika Anda tidak tahu berapa kali perulangan harus dilakukan sebelumnya.

Contoh:

while (kondisi) {

// Blok kode yang akan diulang

}

* Loop for: Loop ini digunakan untuk menjalankan perulangan sejumlah tertentu berdasarkan inisialisasi, kondisi, dan langkah iterasi. Ini sangat berguna ketika Anda tahu berapa kali perulangan harus dilakukan.
* Loop do-while: Loop ini hampir mirip dengan loop while, tetapi pernyataan dalam blok akan dijalankan setidaknya satu kali sebelum kondisi diperiksa. Ini berguna jika Anda ingin memastikan bahwa blok kode dijalankan setidaknya sekali.

Contoh

do {

// Blok kode yang akan diulang

} while (kondisi);

* Loop bersarang (nested loop): Anda dapat menggunakan loop dalam loop, seperti loop while, for, atau do-while, di dalam loop lainnya. Ini digunakan untuk kasus di mana Anda perlu melakukan perulangan bersarang, misalnya, untuk mengakses elemen dalam array multi-dimensi atau untuk membuat pola tertentu.

Contoh

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

// Blok kode yang akan diulang

}

}

Loop adalah salah satu konsep dasar dalam pemrograman yang memungkinkan Anda mengotomatisasi tugas berulang dan mengatasi berbagai jenis permasalahan.

Contoh pemakain loop for berkalang untuk membuat table perkalian

#include <stdio.h>

#define MAKS 8

int main() {

int baris, kolam , hasil\_kali;

for(baris = 1; baris <= MAKS; baris++)

{

for ( kolam = 1; kolam <= MAKS; kolam++)

{

hasil\_kali = baris \* kolam;

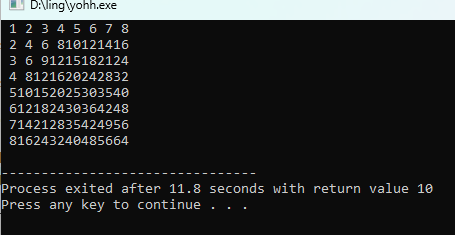
printf("%2d", hasil\_kali);

}

printf("\n");

}

}



**LOOP YANG TAK PERNAH BERHENTI.**

**LOOP** yang tidak pernah berhenti ( berputar terus ) sering terjadi disebabkan adanya kesalahan pada penanganan kondisi yang dipakai untuk keluar dari **loop.**

Sebagai contoh jika ditulis

for ( bil = 60; bil >= 10; bil -= 0)

printf("%d\n", bil );

ternyata pernyataan ini tak akan pernah berhenti dalam menampilkan nilai 60. Kesalahan yang terjadi disebabkan ungkapan ketiga dalam **for** salah tulis. Seharusnya berupa

**Bilangan -= 10**

Namun yang tertulis adalah

**Bilangan -= 0**

Oleh kondisi bilangan **>=** 10 selalu bernilai benar ( sebab bilangan tetap bernilai 60), maka pernyataan

**Printf(“ %d\n ” , bil ) ;**

Dijalankan terus-menerus.

Kalau terjadi hal semacam di atas, tombol CTRL-BREAK dapat dipakai untuk menghentikan eksekusi program

#include <stdio.h>

int main() {

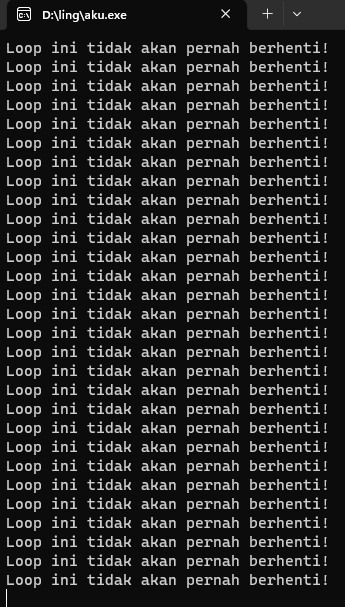
while (1) {

printf("Loop ini tidak akan pernah berhenti!\n");

}

return 0;

}



Program di atas menggunakan pernyataan while (1) yang akan selalu bernilai benar (true), sehingga loop ini akan berjalan selamanya. Anda harus menghentikan program ini secara paksa dengan menghentikan proses atau mematikan aplikasi secara keseluruhan jika ingin menghentikannya. Sebaiknya hindari membuat program semacam ini kecuali Anda benar-benar memahami dampaknya dan membutuhkannya untuk tujuan tertentu.

**PERNYATAAN BREAK**

Pernyataan **break** sesungguhnya telah di perkenalkan pada pernyataan **switch**. Pernyataan ini berfungsi untuk keluar dari **loop for, do-while** dan **while**. Sedangkan pada **switch** yaitu untuk menuju ke akhir (keluar dari )struktur **switch.** Sebagai contoh dapat dilihat pada program. kalau pernyataan break dijalankan maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir tubuh **loop for.**

**for ( ; ; ){**

**.**

**:**

**if (...)**

**break;**

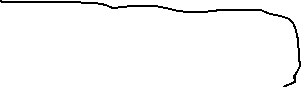
**.**

**:**

**}**



**puts("\nSelesai...");**



akhiran tubuh loop for

pada contoh program berikut, pembaca terhadap tombol akan berakhir kalau tombol akan berakhir kalau tombol yang ditekan adalah ENTER. Pernyataan yang digunakan untuk keperluan ini ;

if ( karakter == ENTER)

break; // keluar dari loop for

yang menyatakan “ jika karakter berupa ENTER, maka keluarlah dari loop for”.

Contoh penggunaan break pada loop pernyataan for

#include <stdio.h>

#include <conio.h>

#define ENTER '\r'

main (){

char karakter;

puts("Anda bisa mengetik apa saja ");

puts("sampai tombol enter anda tekan");

for ( ; ; ){

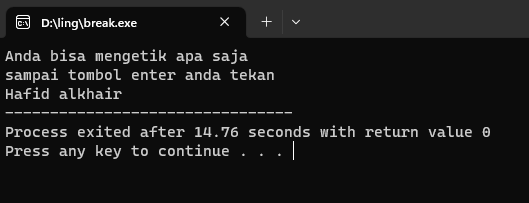
karakter=getche();

if ( karakter == ENTER)

break;

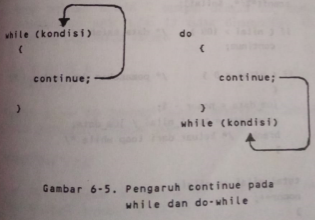
}

}



**PERNYATAAN CONTINUE**

Pernyataan continue digunakan untuk mengarahkan eksekusi ke iterasi ( Proses ) berikutnya pada loop yang sama. Pada do-while, pernyataan continue menyebabkan eksekusi menuju ke kondisi pengujian pengulangan.



Pada while (1)

Menyatakan bahwa kondisi selalu dianggap benar. Untuk keluar dari **loop,** pernyataan digunakan berupa **break.**

**//Contoh pemakain continue pada while**

**#include <stdio.h>**

**main(){**

**int jum\_data =0;**

**int nomor = 1;**

**float tn= 0;**

**float n, r;**

**puts("Untuk mengakhiri pemasukan data");**

**puts("masuk NILAI NEGATIF \n");**

**while (1){**

**printf("data ke-%d :", nomor);**

**scanf("%f", &n);**

**if ( n > 100 )**

**continue;**

**if ( n < 0){**

**jum\_data = nomor - 1;**

**r = tn / jum\_data ;**

**break;**

**}**

**tn += n;**

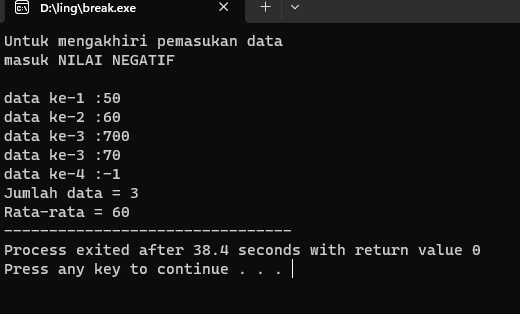
**nomor++;**

**}**

**printf("Jumlah data = %d\n", jum\_data);**

**printf("Rata-rata = %g", r);**

**}**

****

Pada contoh diatas terlihat bahwa saat pemasukan data ke-3 berupa nilai 700, data ini ditolak dan program di minta data masukan kembali.

Pengaruh **continue** pada **loop for** diperhatikan di program dibawah .

Program ini dipakai untuk menampilkan bilangan ganjil yang terletak antara 7 sampai dengan 25, kecuali 15. Untuk menghindari agar nilai 15 tidak tampil ke layer, pernyataan yang digunakan berupa

If ( x == 15)

Continue

Jika kondisi x == 15 bernilai benar, pernyataan continue menyebabkan pernyataan sisanya yaitu

Printf(“%d” , x );

Diabaikan dan eksekusi diarahkan ke ungkapan x += 2 dan kemudian menguji kondisi x <= 25.

CONTOH PEMAKAIN CONTINUE PADA FOR

**#include <stdio.h>**

**main (){**

**int x;**

**for(x =7; x <= 25; x += 2){**

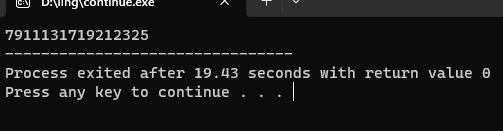
**if ( x == 15)**

**continue;**

**printf("%d",x);**

**}**

**}**



Pada program diatas , pernyataan :

for(x =7; x <= 25; x += 2){

if ( x == 15)

continue;

printf("%d",x);

}

Dapat ditulis dalam bentuk lain sebagai berikut:

for(x =7; x <= 25; x += 2){

if ( x != 15)

continue;

printf("%d",x);

}

PERNYATAAN GOTO.

Pernyataan goto merupakan intruksi untuk mengarahkan eksekusi ke pernyataan yang di awali dengan suatu label. Label sendiri berupa suatu pengenal ( identifier ) yang diikuti dengan tanda titik dua (:).

Contoh pemakain **goto** ditunjukkan pada program bawah.

Pernyataan :

**Goto cetak;**

Megisyratkan agar eksekusi dialnjutkan ke pernyataan yang di awali dengan label

**Cetak** :

Pernyataan

If ( ++ pencacah <= 10 )

Goto cetak;

Mempunyai arti :

* Naikan nilai pencacah besar 1
* Kemudian, jika pencacah kurang dari atau sama dengan 10 maka eksekusi menuju label **cetak**.

#include <stdio.h>

int main() {

int i = 0;

loop:

printf("Perulangan ke-%d\n", i);

i++;

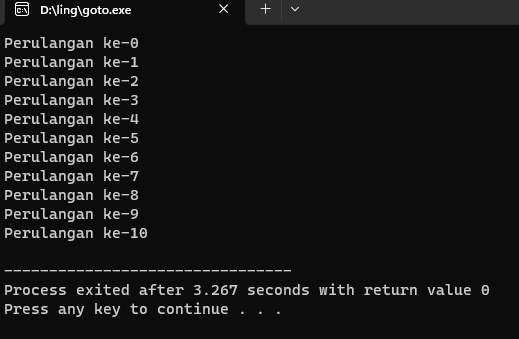
if (i <= 10) {

goto loop; // Loncat kembali ke label 'loop'

}

return 0;

}



Dalam contoh di atas, program akan mencetak "Perulangan ke-X" hingga nilai i mencapai 10. Ketika i mencapai 11, program akan keluar. Penggunaan goto di sini digunakan untuk melompat kembali ke label loop setelah setiap perulangan. Penting untuk diingat bahwa penggunaan goto dalam kode sebaiknya dihindari karena dapat membuat kode menjadi sulit dibaca dan dikelola. Sebaiknya gunakan struktur kontrol seperti while, for, atau do-while untuk membuat loop yang lebih terstruktur dan mudah dipahami.

MENGGUNAKAN EXIT() UNTUK MENGHENTIKAN EKSEKUSI PROGRAM

Suatu ekseskusi program dapat dihentikan melalui pemanggilan fungsi exit(). Hal ini bisa dilakukan, jika di dalam suatu eksekusi terdapat suatu program permainan (games) memerlukan adapter CGA, tetapi yang ada hanyalah monochrome. Oleh karena permainan tak bisa dilakukan maka eksekusi program segera di hentikan.

Menurut kebiasaan, nilai nol diberikan pada argumen exit() untuk menunjukan penghentian program yang normal. Sedangkan untuk menunjukan kesalahan, nilai yang diberikan pada argumen fungsi diisi dengan nilai bukan-nol

Contoh :

If (mono)

Exit();

Pernytaan diatas hanyalah untuk memberikan gambaran yakni jika adapter video adalah monochrome, maka eksekusi program akan dihentikan .

Pada contoh program berikut, eksekusi program akan dihentikan hanya jika tombol ESC ditekan.

#include <stdio.h>

#include <stdlib.h> // Diperlukan untuk menggunakan exit

int main() {

int kondisi = 1; // Ganti dengan kondisi yang sesuai

if (kondisi == 1) {

printf("Kondisi sudah benar. Program akan berhenti.\n");

exit(0); // Keluar dari program dengan status 0 (tidak ada kesalahan)

} else {

printf("Kondisi masih salah. Program akan melanjutkan.\n");

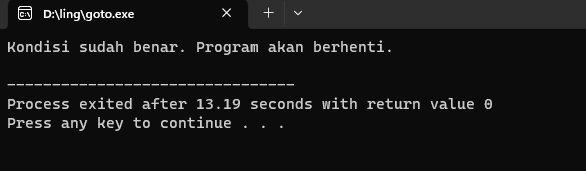
}

// Kode di sini hanya akan dieksekusi jika kondisi == 0

printf("Program melanjutkan eksekusi.\n");

return 0;

}



Dalam contoh di atas, ketika kondisi sama dengan 1, program akan mencetak pesan dan kemudian keluar dari program dengan exit(0). Status 0 menunjukkan bahwa program keluar tanpa kesalahan. Jika kondisi tidak sama dengan 1, program akan mencetak pesan lain dan melanjutkan eksekusi ke bagian selanjutnya. Pastikan untuk memanggil exit hanya saat diperlukan, seperti dalam situasi di mana program harus berhenti secara tiba-tiba karena kesalahan yang serius atau ketika program telah menyelesaikan tugasnya. Jika Anda terlalu sering menggunakan exit dalam kode Anda, ini dapat menyulitkan pemeliharaan dan pengujian program.

Menggabungkan loop for, while, do-while, dan if untuk menghitung jumlah bilangan genap dari 1 hingga N, di mana N adalah angka yang dimasukkan oleh pengguna:

#include <stdio.h>

int main() {

int N;

int total = 0;

printf("Masukkan sebuah angka positif: ");

scanf("%d", &N);

if (N <= 0) {

printf("Angka harus positif.\n");

} else {

// Menggunakan for loop untuk menghitung jumlah bilangan genap dari 1 hingga N

for (int i = 2; i <= N; i += 2) {

total += i;

}

// Menggunakan while loop untuk mencetak bilangan genap dari 1 hingga N

int j = 2;

while (j <= N) {

printf("%d ", j);

j += 2;

}

printf("\n");

// Menggunakan do-while loop untuk mencetak total jumlah bilangan genap

int k = 2;

int sum = 0;

do {

sum += k;

k += 2;

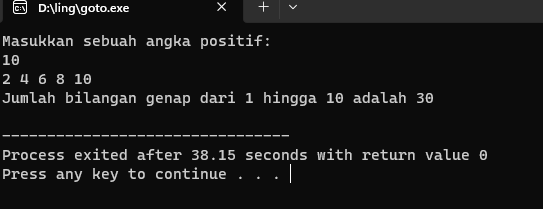
} while (k <= N);

printf("Jumlah bilangan genap dari 1 hingga %d adalah %d\n", N, total);

}

return 0;

}



Program ini meminta pengguna untuk memasukkan angka positif (N), kemudian menggunakan loop for untuk menghitung jumlah bilangan genap dari 1 hingga N, loop while untuk mencetak bilangan genap, dan loop do-while untuk mencetak total jumlah bilangan genap. Program juga mengandung pernyataan if untuk memeriksa apakah angka yang dimasukkan oleh pengguna adalah positif.

Deklarasi variabel biasanya tidak dilakukan dalam loop do-while atau if, tetapi di luar loop atau kondisi tersebut. Namun, Anda dapat mendeklarasikan variabel di dalam blok do-while atau if jika diperlukan. Berikut adalah contoh penggunaannya:Deklarasi dalam Blok do-while:

#include <stdio.h>

int main() {

int angka;

do {

int variabelDalamDoWhile;

printf("Masukkan angka (0 untuk keluar): ");

scanf("%d", &angka);

if (angka != 0) {

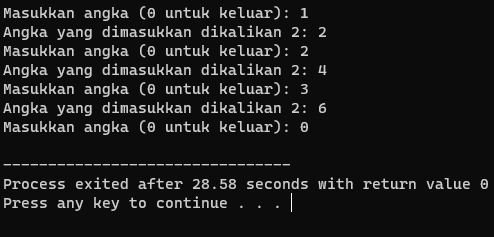
variabelDalamDoWhile = angka \* 2;

printf("Angka yang dimasukkan dikalikan 2: %d\n", variabelDalamDoWhile);

}

} while (angka != 0);

return 0;

}

Dalam contoh di atas, variabel variabelDalamDoWhile dideklarasikan di dalam blok do-while dan digunakan di dalamnya.

Deklarasi dalam Blok if:

#include <stdio.h>

int main() {

int angka = 10;

if (angka > 5) {

int variabelDalamIf; // Variabel dideklarasikan di dalam blok if

variabelDalamIf = angka + 5; // Variabel digunakan di dalam if

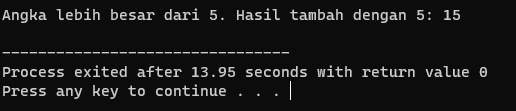
printf("Angka lebih besar dari 5. Hasil tambah dengan 5: %d\n", variabelDalamIf);

}

// variabelDalamIf tidak bisa digunakan di luar dari blok if

// printf("Variabel di luar blok if: %d\n", variabelDalamIf); // Ini akan menghasilkan error

return 0;

}

Dalam contoh di atas, variabel variabelDalamIf dideklarasikan di dalam blok if dan hanya dapat digunakan di dalam blok tersebut. Jika Anda mencoba menggunakannya di luar blok if, akan menghasilkan kesalahan kompilasi.

DAFTAR PUSTAKA

[*https://www.petanikode.com/c-perulangan/*](https://www.petanikode.com/c-perulangan/)

[*https://www.duniailkom.com/tutorial-belajar-c-perulangan-for-bahasa-c/*](https://www.duniailkom.com/tutorial-belajar-c-perulangan-for-bahasa-c/)

[*https://socs.binus.ac.id/2017/03/20/perulangan-dalam-bahasa-c/*](https://socs.binus.ac.id/2017/03/20/perulangan-dalam-bahasa-c/)

[*https://www.sobatambisius.com/2021/09/belajar-bahasa-c-5-perulanganlooping.html*](https://www.sobatambisius.com/2021/09/belajar-bahasa-c-5-perulanganlooping.html)

[*https://www.sobatambisius.com/2021/09/belajar-bahasa-c-5-perulanganlooping.html*](https://www.sobatambisius.com/2021/09/belajar-bahasa-c-5-perulanganlooping.html)

[*https://www.studocu.com/id/document/universitas-muhammadiyah-malang/pemrograman-dasar/perulangan-bahasa-c/34600065*](https://www.studocu.com/id/document/universitas-muhammadiyah-malang/pemrograman-dasar/perulangan-bahasa-c/34600065)

[*https://id.wikibooks.org/wiki/Pemrograman\_C/Perulangan*](https://id.wikibooks.org/wiki/Pemrograman_C/Perulangan)