- Static Variable

  Static variable atau variabel statis adalah jenis variabel yang mempertahankan nilainya. Cara mendeklarasi variabel statis adalah dengan menambahkan syntax static pada inisialisasi variabel.

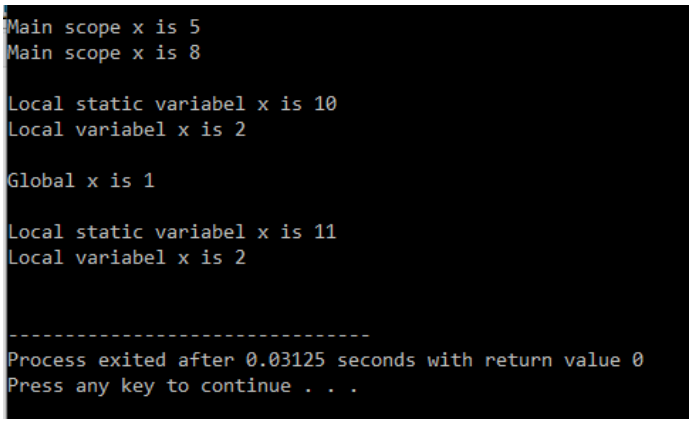
  `static int x;`

  Perhatikan source code berikut :

```
#include "stdio.h"

int x=1; //global variable
void localWithStatic(); //module
void localWoStatic(); //module
void global(); //module

int main(){
int x = 5; //variabel yang hanya dapat digunakan pada fungsi main
printf("Main scope x is %d\n",x); //print variabel x sebelumnya
x+=3;
printf("Main scope x is %d\n\n",x); //print variabel x sebelumnya
localWithStatic(); //memanggil module
localWoStatic(); //memanggil module
global(); //memanggil module
localWithStatic(); //memanggil module
localWoStatic(); //memanggil module
      Return 0;
}

void localWithStatic(){
static int x = 10; //deklarasi variabel statis
printf("Local static variabel x is %d\n",x);
x++; //increment variabel statis x. Nilai x sekarang adalah 11
}

void localWoStatic(){
     //deklarasi variabel yang hanya dapat digunakan pada scope ini
int x = 2;

printf("Local variabel x is %d\n\n",x); //print variabel x
x++; //increment x. Nilai x ssat ini adalah 3
    //ketika module ini dipanggil kembali nilai x kembali seperti semula
}

void global(){
printf("Global x is %d\n\n",x); //print variabel global
}
```

  Jika diamati, terdapat global variabel yang diberi nama `x`. Variabel ini nantinya dapat digunakan pada scope/lingkup manapun. Selanjutnya terdapat 3 buah modul yaitu `localWithStatic()`, `localWoStatic()`, dan `global()`.

1. Module `localWithStatic()` mengilustrasikan penggunaan variabel statis. Diawal module, kita mendeklarasi sebuah variabel statis yang diberi nama `x` dan kita assign dengan nilai 10. Perlu diperhatikan, walau memiliki nama yang sama dengan variabel global, namun variabel `x` pada module ini berbeda dengan variabel global. Berikutnya adalah menampilkan nilai dari variabel statis yang telah didefinisikan sebelumnya. Maka nilai yang akan ditampilkan adalah 10. Diakhir module, variabel `x` kita increment dengan demikian nilai `x` saat ini adalah 11 dan jika module ini dipanggil kembali maka nilai yang ditampilkan adalah 11. Karena sifat dari variabel statis yang mempertahankan nilai.
2. Module `localWoStatic()` mengilustrasikan penggunaan local variable yang artinya adalah variabel yang terdapat pada suatu module hanya dapat digunakan pada module yang memuatnya. Sama halnya dengan module `localWithStatic()` (yang telah dijelaskan pada bagian sebelumnya), variabel yang didefiniskan pada module ini juga diberi nama `x` yang kemudian di-assign dengan nilai 2. Selanjutnya adalah menampilkan nilai dari varibel `x` maka yang akan terpampang dalam layar adalah 2. Module ini diakhiri dengan melakukan increment pada variabel `x` sehingga nilai `x` telah berubah menjadi 3 dan jika module ini dipanggil kembali, nilai yang akan ditampilkan akan kembali seperti semula yaitu 2.
3. Module `global()` mengilustrasikan penggunaan global variabel. Seperti yang telah disebutkan, bahwa variabel global dapat diakses dari lingkup manapun tak terkecuali dalam suatu module yang terpisah sekalipun. Variabel global `x` yang telah didefiniskan di awal program memuat nilai 1 dan jika variabel tersebut di-print melalui module ini maka nilai yang akan ditampilkan adalah 1.



- Register Variable

  Seluruh variabel yang didefinisikan pada suatu program, umumnya akan ditampung dalam memory yang masing-masing memiliki lokasi atau alamat. Sedangkan register variabel adalah jenis variabel yang ditampung ke dalam register komputer. Register dalam

komputer adalah jenis memori pada komputer yang merupakan bagian dari processor. Menyimpan data ke dalam register memiliki keuntungan yaitu proses pemanggilan data lebih cepat. Waktu terbaik untuk menggunakan variabel jenis ini adalah ketika program menjadi lebih kompleks. Berikut adalah syntax dari register variabel :

```
register int x;
```

- External Variable

Ketika program menjadi lebih kompleks, langkah yang tepat adalah memecah beberapa sub-program ke dalam modul-modul kecil. Modul tidak terbatas pada function atau procedure dari file yang sama tetapi module juga dapat diletakkan ke dalam beberapa file terpisah. Mengakses variabel atau bahkan function dari file lain membutuhkan cara khusus. Dalam pemrograman C, cara yang paling mudah adalah menggunakan perintah `extern`. Perintah ini memungkinkan kita untuk mengkases nilai dari suatu variabel atau mengakses function yang terdapat pada file lain. Perhatikan potongan code berikut :

Main.cpp

```
#include "stdio.h"
#include "second.cpp" //linker digunakan untuk menghubungkan kedua
file

extern int x; //mengambil nilai variabel x pada file second.cpp
extern sum(int a, int b); //mengakses function pada file second.cpp

int main(){
//    extern int x;
//    extern sum(int a, int b);

printf("Load data dari luar file : %d\n",x);
printf("%d",sum(x,2));
      return 0;
}
```

Second.cpp

```
int x = 100;
int sum(int a, int b){
return a + b;
}
```

```
Load data dari luar file : 100
102
--------------------------------
Process exited after 0.04264 seconds with return value 0
Press any key to continue . . .
```

Terdapat dua file berektensi .cpp yaitu main.cpp dan second.cpp. Main.cpp adalah file yang memuat fungsi utama untuk menjalankan program sedangkan second.cpp akan diposisikan sebagai modul. Menghubungkan kedua file dapat menggunakan *linker* yang

ditambahkan pada header dari file main.cpp (karena memuat fungsi utama program). Sebagai catatan, menulis struktur `int main(){}` pada file selain file utama akan menyebabkan error. Selain itu, pastikan variabel pada file second.cpp bukan variabel berjenis statis.

Setelah menghubungkan kedua file, langkah selanjutnya adalah mendapatkan nilai dari variabel `x` dan function `sum()` yang terdapat pada file scond.cpp. Pada contoh di atas, variabel eksternal `x` dan fungsi `sum()` akan bersifat global. Variabel eksternal juga dapat dideklarasi pada suatu scope. Pada contoh di atas, terdapat syntax untuk mendeklarasi variabel dan fungsi eksternal di dalam fungsi utama (sebelum perintah print).

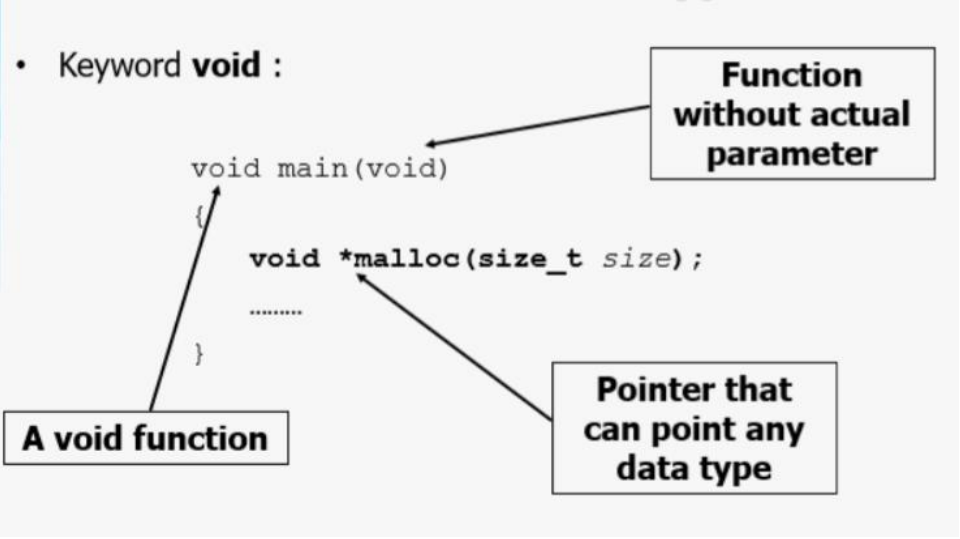
- Tipe Data Void

  Void dalam dunia programming digolongkan ke dalam tipe data khusus yang tidak memiliki data. Gambar di bawah ini menunjukkan bahwa void dapat digunakan pada fungsi yang tidak mengembalikan nilai atau biasa disebut procedure. Selain itu, void juga bisa digunakan pada parameter sebuah fungsi. Jika kita menuliskan void pada parameter fungsi maka fungsi tersebut tidak memiliki parameter. Perhatikan potongan code berikut

  ```
  Int someFunction(){return someValue;}
  ```

  bisa juga dituliskan seperti berikut

  ```
  int someFunction(void){return someValue;}
  ```



- Command Line Argument

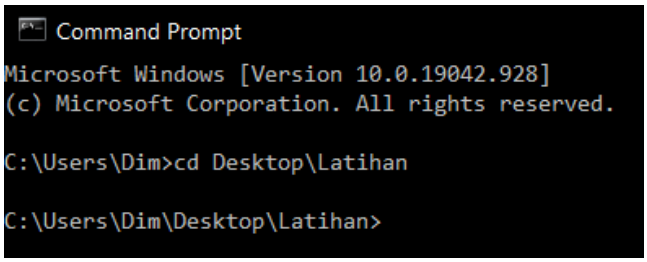
  Selama ini, kita menjalankan program C menggunakan aplikasi compiler seperti Dev-C++. Pada beberapa sistem operasi, program C bisa langsung dijalankan tanpa menggunakan aplikasi compiler tambahan. Kita cukup menggunakan command line/terminal/CMD

untuk mengeksekusi program C. Pada sistem operasi windows dibutuhkan *tool* tambahan yaitu MingW agar program C dapat dieksekusi melalui CMD. Cara instal dan konfigurasi MingW dapat diakses melalui https://www.wikihow.com/Run-C-Program-in-Command-Prompt. Perhatikan code berikut :

Hello_world.cpp

```
#include "stdio.h"

int main(){
printf("Hello world");
return 0;
}
```

Code di atas akan menampilkan pesan “Hello world” dan untuk mengeksekusi program tersebut buka CMD/terminal pada komputer lalu arahkan pada directory atau lokasi program tersimpan. Pada contoh ini, program saya simpan di folder latihan yang terdapat pada *Desktop.* Berikut adalah command yang perlu dituliskan untuk mengarahkan CMD ke directory yang dimaksud.

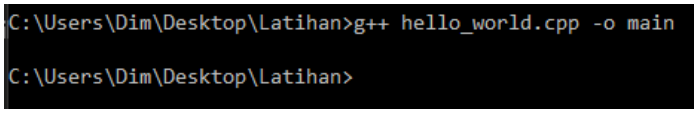


Keterangan :

Cd (change directory) adalah perintah untuk berpindah directory

Desktop\Latihan adalah lokasi yang dituju

Jika berhail, maka akan seperti pada baris berikutnya. Langkah selanjutnya adalah me-*compile* program hello_world.cpp yang telah dibuat. Pastikan file nama_program.cpp telah tersedia pada directory yang dituju. Ketikkan perintah berikut pada CMD untuk melakukan compile program.



Keterangan :

g++ adalah perintah untuk melakukan compile *program*

Hello_world.cpp adalah nama file program C

Main adalah nama executable program

Jika perintah di atas berhasil dijalankan maka akan tercipta sebuah executable file baru dan pada contoh ini executable file yang tercipta adalah main.exe. Terakhir adalah perintah untuk mengeksekusi program. Berikut adalah perintah yang perlu ditulis `.\nama_executable_program`.

```
C:\Users\Dim\Desktop\Latihan>.\main
Hello world
C:\Users\Dim\Desktop\Latihan>
```

Berikutnya akan dijelaskan cara menggunakan command line argument. Command line argument memungkinkan kita untuk mengirim argument kepada fungsi utama program. Sekilas hampir sama dengan mengirim argument kepada fungsi lain namun yang menjadi pembeda adalah, pada command line argument, argument akan dikirim ke fungsi utama tepat pada saat program dijalankan sedangkan passing argument yang biasa akan dikirim pada suatu fungsi ketika program telah dijalankan. Perhatikan code berikut :

Hello_world.cpp

```
#include "stdio.h"

int main(int argc, char *argv[]){ //parameter yang diterima oleh
fungsi utama

      printf("Hello %s\n",argv[1]);
return 0;
}
```

Jika diamati, terdapat dua buah parameter yang akan diterima oleh fungsi utama, yaitu `int argc` dan `char **argv`. Parameter `int argc` akan menerima atau memuat jumlah argument yang dikirimkan sedangkan `char **argv` akan menampung argument yang dikirim. Program di atas akan menampilkan pesan “Hello argument_yang_diterima”. Untuk mengeksekusi program hal yang perlu dilakukan adalah melakukan compile program menggunakan perintah `g++ hello_world.cpp -o main` pada CMD. Lalu ketikkan perintah berikut `.\nama_executable_program “argument”`

```
C:\Users\Dim\Desktop\Latihan>.\main Dimas
Hello Dimas

C:\Users\Dim\Desktop\Latihan>
```