Free (Deallocate Memory Block) adalah salah satu fungsi untuk membebaskan atau menghapus memori, yang tersedia pada bahasa pemrograman C dan Juga pada bahasa pemrograman C++. Free tersedia pada header <stdlib.h> / <cstdlib>.

Free berfungsi untuk menghapus atau membebaskan memori yang sudah dialokasi sebelumnya dengan menggunakan fungsi allocator seperti malloc, new, calloc, realloc dan lain-lain. Menghapus memori atau membebaskan memori jika sudah tidak dibutuhkan lagi, untuk dikembalikan kepada heap dan memori tersebut akan tersedia untuk alokasi berikutnya.

Setelah melakukan pemesanan atau alokasi memori kita sangat disarankan untuk membebaskan atau menghapus memori yang telah kita pesan jika memang sudah tidak akan digunakan kembali, Hal ini berguna untuk mencegah adanya kebocoran memori (Memory Leak).

Jika pointer bukan menunjuk alamat memori yang berasal dari operasi fungsi untuk alokasi memori Dan jika pointer tersebut NULL, maka fungsi Free tidak melakukan apapun.

Dalam pembebasan atau penghapusan memori memang tidak membuat pointer tersebut bernilai null atau berhenti untuk menunjuk memori tersebut. Pada saat melakukan alokasi atau pemesanan, C++ meminta kepada Opertating System untuk mencarikan memori yang dapat digunakan, setelah mendapatkanya memori tersebut akan diserahkan ke pointer dan ditandai sebagai memori yang sudah dialokasi atau memori yang sedang dipakai, dan saat pembebasan atau penghapusan memori tersebut akan dikembalikan sebagai memori yang dapat di pakai untuk alokasi memori selanjutnya, itu alasan kuat untuk mewajibkan anda membebaskan setiap memori yang sudah dipesan jika sudah tidak dperlukan kembali. Dan juga setelah dibebaskan pointer tidak secara otomatis menjadi null. Pointer masih menunjuk ke memori tersebut, kita masih dapat mengaksesnya bahkan mengubah nilai memori tersebut tetapi ingat bahwa memori tersebut sudah bukan milik anda. Kemungkinan anda bisa mengubah nilai memori milik program lain jika memori itu sudah mendapatkan pemiliknya yang baru. Untuk mencegah hal tersebut kita bisa membuat pointer tersebut menjadi null langsung setelah pembebasan memori.

Bentuk umum penulisan / Syntax :

void free (void\*ptr) ;

Parameter :

Ptr = pointer berisi alamat memori yang akan dibebaskan atau dihapus.

Contoh Penulisan :

free (pointer) ;

free (varFoo);

Contoh Program :

#include <stdlib.h>

#include <iostream>

#define NEWLINE cout<<endl; //Konstanta

using namespace std;

int main (){

int \*ptrMall = nullptr, \*ptrCall = nullptr, \*ptrReall = nullptr ; //Null Pointer

cout<<"ptrMall = "<<ptrMall<<endl; //Null

cout<<"ptrCall = "<<ptrCall<<endl; //NULL

cout<<"ptrReall = "<<ptrReall<<endl; //NULL

NEWLINE

ptrMall = (int \*) malloc(sizeof(int));

ptrCall = (int \*) calloc(1, sizeof(int));

cout<<"ptrMall = "<<ptrMall<<" = "<<\*ptrMall<<endl;

cout<<"ptrCall = "<<ptrCall<<" = "<<\*ptrCall<<endl; //NULL

cout<<"ptrReall = "<<ptrReall<<endl; //NULL

NEWLINE

ptrReall = (int\*) realloc(ptrMall, 2\*sizeof(int));

cout<<"ptrMall = "<<ptrMall<<" = "<<\*ptrMall<<endl;

cout<<"ptrCall = "<<ptrCall<<" = "<<\*ptrCall<<endl; //NULL

cout<<"ptrReall = "<<ptrReall<<" = "<<\*ptrReall<<endl; //NULL

NEWLINE

//free (ptrMall); //Sudah Dibebaskan saat pemanggilan realloc

free (ptrCall); //Hapus

free (ptrReall); //Hapus

cout<<"ptrMall = "<<ptrMall<<" = "<<\*ptrMall<<endl;

cout<<"ptrCall = "<<ptrCall<<" = "<<\*ptrCall<<endl; //NULL

cout<<"ptrReall = "<<ptrReall<<" = "<<\*ptrReall<<endl; //NULL

NEWLINE

ptrMall = nullptr;

ptrCall = nullptr;

ptrReall = nullptr;

cout<<"ptrMall = "<<ptrMall<<endl; //Null

cout<<"ptrCall = "<<ptrCall<<endl; //NULL

cout<<"ptrReall = "<<ptrReall<<endl; //NULL

return 0;

}

Dari contoh di atas sudah sangat jelas, Contoh diatas akan menunjukan cara penggunaan dan melihat tahap demi tahap perubahan apa yang terjadi.