**Friend Function**

Friend Function adalah keyword yang membuat function menjadi spesial yang memungkinkan kita untuk dapat mengakses member dari class yang bersifat private atau protected (member yang tidak bisa di ases dari luar class). Sebelumnya dijelaskan bahwa terdapat tiga akses specifier yaitu public, private dan protected yang dengan sifat mereka masin-masing. Terdapat dua akses specifier yang mempunyai sifat tidak bisa di akses dari luar class.

Dengan mendirikan friend function di dalam class sebagai member public memungkinan kita untuk mengakses member yang bersifat private dari class tersebut. friend function bukan anggota/member dari class tersebut meskipun didrikan di dalam class. Untuk membuat friend function dibutuhkan keyword “friend” di depan deklarasi function.

Bentuk penulisan

class nama\_class{

public

friend void nama\_function(nama\_class nama\_parameter){

//Definisi

}

}

Contoh penulisan

class siswa{

private:

string nama;

public:

friend void setNama(siswa a,string pnama){

a.nama = pnama;

}

};

Di atas adalah contoh penulisan dari friend function, pada parameter tersebut terdapat parameter yang mempunyai tipe data class siswa itu digunakan untuk melakukan akses kepada member object yang dimaksud. Ingat bahwa kemungkinan seorang membuat friend function digunakan untuk mengakses member dari class. Dalam class terdapat dua jenis member yaitu static dan non-static member. Static member yang bersifat akan tersedia untuk object dan di luar object (member milik class) dan non-static member hanya akan ada di dalam object itu sendiri (hanya dapat di akses melalui object). jadi kita membutuhkan object yang agar dapat mengakses non-static member.

Contoh program

#include <iostream>

#include <string>

using namespace std;

class siswa{

private:

string nama;

public:

friend void setNama(siswa &a,string);

};

void setNama(siswa &a, string pNama){

a.nama = pNama;

}

int main(){

siswa budi;

setNama(budi,"Budi Komaa");

return 0;

}

**Friend Class**

Friend untuk class memungkinkan kita mendapatkan akses ke member class lain yang bersifat private atau protected. Masih sama seperti friend function, untuk melakukanya kita membutuhkan keyword “friend” dan selanjutnya nama dari class. Friend class akan berlaku untuk class yang dipanggil oleh class yang memanggilnya, membuat class yang dipanggil dapat mengakses semua member dari class yang memanggil walaupun bersifat private atau protected.

Bentuk penulisan

class nama\_class{

friend class nama\_class2;

};

Contoh program

#include <iostream>

#include <string>

using namespace std;

class siswa;

class orang{

private:

string nama;

public:

void setNama(string pNama);

friend class siswa;

};

class siswa{

private:

int id;

public:

void setId(int pId);

void displayAll(orang &a);

};

void siswa::displayAll(orang &a){

cout<<id<<endl<<a.nama;

}

void orang::setNama(string pNama){

nama = pNama;

}

void siswa::setId(int pId){

id=pId;

}

int main(){

orang budi;

budi.setNama("Budi Komaa");

siswa budia;

budia.setId(1);

budia.displayAll(budi);

return 0;

}

Di atas adalah contoh program dari penggunaan keyword friend, yang digunakan untuk menampilkan keluaran nilai member dari dua object dan dua class. pada awal sebelum class “orang” didirikan class prototype, yang menyatakan bahwa class “siswa” akan di dirikan dan anda bisa memanggilnya meskipun definisi class tersebut berada di bawah pemanggilan.

Friend class “siswa” tersebut didirikan di dalam class” orang”, yang berarti class “siswa” dapat mengakses member private dari class “orang”. Tapi kita tetap akan membutuhkan object karena apa yang di akses adalah non-static member.

**Multiple Friend**

Keyword friend dapat digunakan di beberapa class dengan nama yang sama pada waktu yang bersamaan. Hal itu menghasilkan function atau class yang dianggap sebagai friend tersebut dapat mengakses member object yang bersifat private di beberapa class (lebih dari satu class).

Contoh program

#include <iostream>

using namespace std;

class jeruk;

class mangga{

private:

int id;

public:

mangga():id(1){} //Default Constructor

friend void printAll(mangga &pA, jeruk &pB);//Friend

};

class jeruk{

private:

int id;

public:

jeruk():id(2){} //Default Constructor

friend void printAll(mangga &pA, jeruk &pB);//Friend

};

void printAll(mangga &pA, jeruk &pB){//Definisi Friend

cout<<"Mangga = "<<pA.id<<endl;

cout<<"Jeruk = "<<pB.id<<endl;

}

int main(){

mangga myMangga;

jeruk myJeruk;

printAll(myMangga, myJeruk);

return 0;

}

Di atas adalah contoh program, dimana function “printAll” merupakan friend dari dua class yaitu “jeruk” dan “mangga”, hal itu akan membuat function “printAll” mendapatkan ijin akses untuk member private atau protected mereka. Hal itu dapat kita lihat pada definisi “printAll” yang dapat melihat nilai dari member variabel “id” dari kedua object. dan pada parameter deklarasi “printAll” menggunakan dua parameter degan tipe data kedua class, ingat bahwa yang kita akses adalah non-static member, non-static member hanya dimiliki oleh object maka dari tu kita butuh object untuk mengakses member tersebut.

Friend Member Function

Function yang dianggap friend oleh suatu class juga bisa merupakan anggota atau member dari class lain. Membuat Member function tersebut memliki ijin atas hak akses member private atau protected yang menganggap member function tersebut sebagai friend.

Contoh program

#include <iostream>

using namespace std;

class mangga; //prototype

class jeruk{

private:

int id;

public:

jeruk():id(2){} //Default Constructor

void printAll(mangga &pA);//Friend

};

class mangga{

private:

int id;

public:

mangga():id(1){} //Default Constructor

friend void jeruk::printAll(mangga &pA);//Friend

};

void jeruk::printAll(mangga &pA){//Definisi Friend

cout<<"Mangga = "<<pA.id<<endl;

cout<<"Jeruk = "<<id<<endl;

}

int main(){

mangga myMangga;

jeruk myJeruk;

myJeruk.printAll(myMangga);

return 0;

}

Contoh program di atas menunjukan bahwa member function “jeruk::printAll” adalah friend dari class “mangga”, yang berarti member function tersebut dapat mengakses member private yang berada pada class “mangga”. Seperti biasanya kita tetap butuh object dari class “mangga” untuk mengakses non-static member pada class mangga.

Di atas class jeruk di dirikan protorype class mangga, menyatakan mangga akan di definisikan setelah jeruk. Kita dirikan prototype di atas karena kita butuh dentifier mangga di dalam parameter “printAll”, dan juga mangga di definisikan di bawah jeruk karena di dalam class tersebut kita dirikan friend function yang menyatakan salah satu member dari “jeruk” yaitu “printAll”.

Friend juga bisa digunakan pada Function Template dan Class Template.