Nama :Martin Caesar Partogi

Kelas :TI-24-PA2

NPM :242310034

Constructor dan Destructor

1. Pengertian Constructor dan Destructor

a. Constructor

Constructor adalah metode khusus dalam sebuah kelas yang dipanggil secara otomatis saat objek dari kelas tersebut dibuat. Fungsinya adalah untuk menginisialisasi atribut atau melakukan setup awal.

Konstruktor (constructor) sendiri merupakan sebuah fungsi yang otomatis akan dipanggil setiap kali melakukan instansiasi terhadap kelas. Dapat diartikan juga suatu fungsi dari anggota suatu kelas yang memiliki nama yang sama dengan nama kelas fungsi itu berada. Konstruktor ini digunakan untuk mengalokasikan ruang untuk suatu objek dan untuk memberikan nilai awal (Stroustrup, 2013). Sama seperti fungsi biasa, constructor dapat ditambahkan parameter maupun dilakukan overload. Ketika membuat constructor pastikan nama dari fungsi constructor harus sama dengan nama kelasnya dan tidak memiliki tipe kembalian (tidak juga void).

Ciri-ciri Constructor:

- Nama constructor sama dengan nama kelas.
- Tidak memiliki tipe pengembalian (return type).
- Bisa memiliki parameter (constructor berparameter) atau tanpa parameter (default constructor).

b. Destructor

Destructor adalah metode khusus dalam sebuah kelas yang dipanggil secara otomatis saat objek dihancurkan atau keluar dari cakupan (scope). Fungsinya adalah untuk membersihkan atau melepaskan sumber daya yang dialokasikan oleh objek.

Destructor merupakan kebalikan dari constructor. Destructor adalah method khusus yang dijalankan saat objek dari sebuah class dihapus atau tidak lagi digunakan. Biasanya fungsi ini dijalankan secara otomatis ketika program telah selesai dijalankan Destructor memiliki nama yang sama dengan nama class dimulai dengan tanda tilde ('~') dan tidak memiliki tipe data atau parameter. Destructor digunakan untuk membersihkan memori dan menjalankan tugas-tugas lain seperti menutup file atau membersihkan data.

Ciri-ciri Destructor:

- Nama destructor sama dengan nama kelas diawali dengan tanda ~.
- Tidak memiliki parameter.
- Tidak mengembalikan nilai.

2. Implementasi Constructor dan Destructor

a. Contoh Implementasi Destructor

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

class pelajaran { // class name
    public: // access specifier
        pelajaran () { // constructor
            cout << "Ini adalah mencari materi C++ tentang constructors!" << endl;
        }
};

int main () {
    pelajaran obj; // membuat object dari sebuah class
    return 0;
}
```

b. Contoh Implementasi Constructor sebagai tempat inisialisasi daya member

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

class laptop {
    private:
        string pemilik;
        string merk;

    public:
        laptop(string var1, string var2) {
            pemilik = var1;
```

```
        merk = var2;
        cout << "Paket laptop " << merk << " milik " << pemilik << " sudah dikirim" <<
endl;
    }
};

int main() {
   laptop laptopFadlan("Fadlan", "Asus");
   laptop laptopRisma("Risma", "Acer");
   laptop laptopCarlos("Carlos", "Lenovo");

   return 0;
}
```

c. implementasi Destructor

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

class mahasiswa {
        public :
                mahasiswa (string nama, int umur)
                        : nama_(nama), umur_(umur){

                         }
                        ~mahasiswa () {
                                cout << "==Contoh Destructor Mahasiswa==" << endl;
                        }

                        void tampilkanbiodata() {
                                cout << "Nama : " << nama_ << endl;
```

```
                cout << "Umur : " << umur_ << endl;
        }

    private :
        string nama_;
        int umur_;
};

int main ()
{
    {
    mahasiswa mhs("Katarina", 20);
    mhs.tampilkanbiodata();
    }
    return 0;
}
```

3. Kesimpulan

Dalam bahasa C, constructor dan destructor tidak tersedia secara otomatis seperti dalam C++, tetapi kita dapat mengimplementasikannya dengan menggunakan fungsi khusus. Constructor digunakan untuk menginisialisasi struktur atau objek dengan cara mengalokasikan memori dan mengatur nilai awal atribut. Ini memastikan bahwa objek siap digunakan dengan benar setelah dibuat.

Sebaliknya, destructor bertanggung jawab untuk membersihkan dan melepaskan sumber daya yang telah digunakan oleh objek. Ini sangat penting dalam manajemen memori manual di bahasa C untuk menghindari kebocoran memori (memory leak), yang dapat menyebabkan penggunaan sumber daya yang tidak efisien dan potensi crash aplikasi.

Dalam aplikasi yang lebih kompleks, penggunaan constructor dan destructor menjadi semakin penting, terutama ketika bekerja dengan struktur data yang dinamis, seperti daftar tertaut, pohon biner, atau sistem berbasis alokasi memori dinamis lainnya. Penggunaan destructor yang tepat dapat memastikan bahwa tidak ada alokasi memori yang tertinggal setelah program selesai dieksekusi.

Selain itu, dalam pengembangan perangkat lunak yang skalabel dan efisien, pengelolaan memori yang baik sangat diperlukan untuk mencegah fragmentasi memori dan memastikan performa optimal. Dengan menerapkan pola constructor dan destructor yang baik, kita dapat meningkatkan keamanan, efisiensi, dan keandalan program yang ditulis dalam bahasa C.

Kesimpulannya, meskipun bahasa C tidak memiliki fitur constructor dan destructor secara langsung seperti dalam C++, kita masih dapat mengimplementasikannya secara manual dengan menggunakan fungsi khusus. Dengan memahami konsep ini dan menggunakannya dengan benar, kita dapat memastikan bahwa program yang kita buat lebih terstruktur, mudah dikelola, dan terbebas dari masalah memori yang sering terjadi dalam pemrograman tingkat rendah.