**Cấp phát động trong C++**

1. **new operator**

Toán tử **new** được sử dụng để cấp phát bộ nhớ động cho một biến, một mảng hoặc một đối tượng.

Khi sử dụng **new**, bộ nhớ sẽ được cấp phát trên phân vùng **Heap** (khu vực nhớ dành cho cấp phát động) thay vì trên stack (khu vực nhớ dành cho cấp phát tĩnh).

**new** tương đương với hàm **malloc(), calloc()** trong C.

1. **delete operator**

Toán tử **delete** dùng để giải phóng bộ nhớ đã được cấp phát bằng **new**. Điều này giúp tránh lãng phí bộ nhớ và hiện tượng rò rỉ bộ nhớ (memory leak).

**delete** tương đương với hàm **free()** trong C.

**Ví dụ 1:**  **Cấp phát động một biến đơn**

int \*ptr = new int(10);

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

     \* new int(10)

     \*      + Khởi tạo một biến int

     \*      + Cấp phát động cho biến int (0x01 - Heap)

     \*      + Gán giá trị tại địa chỉ vừa cấp phát là 10

     \*      + Trả về địa chỉ của vùng nhớ mới được cấp phát

     \*

     \* con trỏ ptr sẽ trỏ đến địa chỉ trên

     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    cout << "Địa chỉ đối tượng: " << ptr << endl;

    cout << "Giá trị đối tượng: " << \*ptr << endl;

    delete ptr;

**Ví dụ 2: Cấp phát động một mảng**

#include <iostream>

using namespace std;

int main(int argc, char const \*argv[])

{

    int size;

    cout << "Nhập kích thước của mảng: ";

    cin >> size;

    int \*arr = new int[size];

    for (int i = 0; i < size; i++){

        arr[i] = i \* 2;

    }

    for (int i = 0; i < size; i++){

        cout << arr[i] << " ";

    }

    cout << endl;

    delete[] arr;  // Giải phóng bộ nhớ

    return 0;

}

* Mặc dù việc cấp phát động bằng new trong C++ không thể thay đổi kích thước giống như realloc trong C, nhưng cấp phát động vẫn có nhiều điểm khác biệt so với khai báo mảng tĩnh.
* **Mảng tĩnh**:
* Kích thước phải được xác định khi biên dịch (compile-time), nghĩa là bạn cần biết trước số lượng phần tử và không thể thay đổi trong khi chạy chương trình.
* Bộ nhớ của mảng tĩnh được cấp phát trên stack, và nó sẽ tự động giải phóng khi ra khỏi phạm vi (scope) của nó. Bộ nhớ mảng tĩnh không tồn tại lâu dài trong suốt chương trình mà phụ thuộc vào phạm vi khai báo.
* Kích thước mảng trong stack bị giới hạn bởi kích thước của stack (thường là vài MB), nên nếu bạn cần một mảng lớn, mảng tĩnh có thể không khả thi và dễ gây lỗi stack overflow.
* **Cấp phát động với new**:
* Bạn có thể xác định kích thước mảng tại thời điểm chạy, giúp linh hoạt hơn nếu kích thước mảng phụ thuộc vào các giá trị mà bạn chỉ biết trong quá trình chạy chương trình.
* Bộ nhớ được cấp phát trên heap, có thể tồn tại cho đến khi bạn giải phóng thủ công bằng **delete[].** Điều này cho phép bộ nhớ tồn tại lâu dài hơn, ngay cả khi phạm vi khai báo của con trỏ đã kết thúc.
* Vì bộ nhớ cấp phát động nằm trên heap, bạn có thể tạo ra các mảng rất lớn nếu bộ nhớ hệ thống còn đủ, mà không bị giới hạn bởi stack.

**Ví dụ 3: Cấp phát động một đối tượng của class**

#include <iostream>

using namespace std;

class SinhVien{

    private:

        string name;

        int id;

        int age;

    public:

        SinhVien(string \_name, int \_age): name(\_name), age(\_age){

            static int \_id = 1;

            id = \_id;

            \_id++;

        }

        void display(){

            cout << "Name: " << name << ", ID: " << id << ", Age: " << age << endl;

        }

};

int main(int argc, char const \*argv[])

{

    SinhVien \*p1 = new SinhVien("Trung", 25);

    p1->display();

    SinhVien \*p2 = new SinhVien("Tuan", 22);

    p2->display();

    return 0;

}

**Ví dụ 4: Quản lý thông tin các hãng xe**

#include <iostream>

#include <string>

using namespace std;

class Xe{

    public:

        virtual void hienThiThongTin() = 0;

};

class Toyota : public Xe{

    private:

        string model;

        int namSanXuat;

        string dongCo;

    public:

        Toyota(string m, int nam, string dongCo): model(m), namSanXuat(nam), dongCo(dongCo){}

        void hienThiThongTin() override {

            cout << "Hang xe: Toyota\n";

            cout << "Model: " << model << "\n";

            cout << "Nam san xuat: " << namSanXuat << "\n";

            cout << "Dong co: " << dongCo << "\n";

        }

};

class Honda : public Xe{

    private:

        string model;

        int namSanXuat;

        string mauSac;

    public:

        Honda(string m, int nam, string mau): model(m), namSanXuat(nam), mauSac(mau){}

        void hienThiThongTin() override {

            cout << "Hang xe: Honda\n";

            cout << "Model: " << model << "\n";

            cout << "Nam san xuat: " << namSanXuat << "\n";

            cout << "Mau sac: " << mauSac << "\n";

        }

};

class Mazda : public Xe{

    private:

        string model;

        int namSanXuat;

        string loaiDanDong;

    public:

        Mazda(string m, int nam, string loaiDanDong): model(m), namSanXuat(nam), loaiDanDong(loaiDanDong){}

        void hienThiThongTin() override {

            cout << "Hang xe: Mazda\n";

            cout << "Model: " << model << "\n";

            cout << "Nam san xuat: " << namSanXuat << "\n";

            cout << "Loai dan dong: " << loaiDanDong << "\n";

        }

};

int main() {

    // Toyota to;

    // Honda ho;

    // Mazda mz;

    // Xe \*xe;

    // xe = &to;

    // xe->hienThiThongTin();

    Mazda cx3("CX-3", 2019, "Dan dong 4 banh");

    Xe \*ptr[] = {new Toyota("campry", 2020, "V6"),

                 new Honda("civic", 2019, "do"),

                 new Mazda("CX-5", 2021, "Dan dong 4 banh"),

                 new Toyota("corolla", 2018, "I4 1.8L"),

                 new Honda("accord", 2020, "den"),

                 &cx3};

    for (int i=0; i<6; i++){

        ptr[i]->hienThiThongTin();

        cout << "--------------" << endl;

        delete ptr[i];

    }

    return 0;

}

**Ví dụ 5: Cấp phát động cho một biến của struct**

#include <iostream>

using namespace std;

typedef struct{

    int x;

    int y;

} Point;

int main(int argc, char const \*argv[])

{

    Point \*p1 = new Point;

    p1->x = 10;

    p1->y = 20;

    cout << "Point 1: (" << p1->x << ", " << p1->y << ")" << endl << endl;

    Point \*p2 = new Point{5, 15};

    cout << "Point 2: (" << p2->x << ", " << p2->y << ")" << endl << endl;

    Point \*points = new Point[3];    // Cấp phát động cho một mảng gồm 3 biến Point

    points[0] = {1,2};

    points[1] = {3,4};

    points[2] = {5,6};

    for (int i = 0; i < 3; i++) {

        cout << "Point " << i + 1 << ": (" << points[i].x << ", " << points[i].y << ")" << endl;

    }

    delete p1;

    delete p2;

    delete[] points;

    return 0;

}