Constructors, Destructor, Operators

Nguyễn Văn Khiết

Nội dung

- Constructors
- Destructors
- Operators
- Luyện tập

Constructors

- Khi vừa được khai báo, đối tượng cần có thông tin khởi tạo ban đầu.
- Có thể dùng một số phương thức để khởi tạo thông tin cho đối tượng. Đôi khi, người dùng quên gọi hàm khởi tạo → đối tượng không có giá trị ban đầu → đôi khi gây lỗi
- → Constructor (hàm khởi tạo, hàm dựng) giúp khởi tạo giá trị của đối tượng ngay khi nó được tạo ra.

Constructors

- Hàm khởi tạo trùng tên với tên lớp.
- Hàm khởi tạo không có giá trị trả về
- Có thể có nhiều hàm khởi tạo với bộ tham số khởi tạo khác nhau
- Các hàm khởi tạo có thể gọi lẫn nhau
- Khi tạo đối tượng (khai báo hoặc new), tùy vào danh sách tham số hàm khởi tạo tương ứng sẽ tự động được gọi.
- Ví dụ.

Constructors

- Default Constructor (hàm khởi tạo mặc định):
  - Khi không có hàm khởi tạo nào tạo ra, xem như tồn tại một hàm khởi tạo mặc định
    - Khởi tạo mặc định các thuộc tính đối tượng.

Copy Constructors

- Copy Constructor (hàm khởi tạo sao chép)
  - Tạo đối tượng có thuộc tính giống một đối tượng có sẵn.
  - Hàm khởi tạo sao chép:
    - Có đúng 1 tham số là đối tượng cùng loại.
    - Dùng để khởi tạo một đối tượng giống đối tượng truyền vào.
    - Khi không có hàm khởi tạo sao chép:
      - Xem như tồn tại một hàm khởi tạo sao chép
      - Hàm này gán từng thuộc tính từ đối tượng tham số truyền vào

Copy Constructors

- Copy Constructor (hàm khởi tạo sao chép) của lớp PhanSo

```
PhanSo::PhanSo(const PhanSo& ps)
{
   TuSo = ps.TuSo;
   MauSo = ps.MauSo;
}
```

Destructor

- Khi tạo ra đối tượng và trong quá trình đối tượng hoạt động:
  - Đối tượng có thể cấp phát bộ nhớ
  - Mở file
  - Mở kết nối đến CSDL, mạng, …
  - Dùng một tài nguyên nào đó
- ➔ Cần giải phóng các tài nguyên trên khi không dùng đối tượng nữa.
- ➔ Tạo một phương thức để hủy
  - Đôi khi người dùng quên gọi

Destructor

- Hàm hủy là hàm được tự động gọi khi đối tượng bị hủy.
- Dùng hàm hủy để giải phóng các vùng nhớ, tài nguyên mà đối tượng đã tạo ra
- ➔ Giải quyết vấn đề quên gọi

Destructor

- Trong C++, hàm hủy có tên trùng với tên lớp, nhưng có thêm dấu ~ ở đầu.
- Hàm hủy không có tham số
- Hàm hủy không có giá trị trả về
- Ví dụ:
  - ~GPS()

Operators

- Trong C++, cho phép việc định nghĩa toán tử trên các lớp → thực hiện các biểu thức tính toán tự nhiên.
- Khi sử dụng đối tượng với toán tử định nghĩa, hàm toán tử tương ứng sẽ được gọi.
- Chỉ có thể định nghĩa một số toán tử cho trước (trong danh mục), không thể định nghĩa các toán tử mới.

Operators

- Trong C++, dùng cú pháp **operator toantu** khi định nghĩa toán tử.
- Viết như một hàm bình thường.
- Ví dụ:

```
PhanSo operator+(const PhanSo&);
////////////////////////////////
PhanSo PhanSo::operator+(const PhanSo& ps)
{
  PhanSo kq;
  kq.TuSo = TuSo * ps.MauSo + ps.TuSo * MauSo;
  kq.MauSo = MauSo * ps.MauSo
  return kq;
}
```

Operators

- Các loại toán tử
  - Toán tử 1 ngôi
    - Tiền tố

      PhanSo& operator++();
    - Hậu tố

      PhanSo operator++(int);
  - Toán tử 2 ngôi

    PhanSo PhanSo::operator+(const PhanSo& ps)

Operators

- Đối với từng toán tử, có thể nạp chồng với nhiều tham số khác nhau
- Ví dụ: Nạp chồng toán tử gán của lớp PhanSo:

  PhanSo operator +(*int number*);

  float operator +(*float number*);

Operators

- Toán tử độc lập: toán tử ở cấp toàn cục và không thuộc lớp nào
- Thông thường, định nghĩa mở rộng thêm ứng xử của toán tử đó với một lớp mới định nghĩa
- Ví dụ:
    - Toán tử <<

Operators

- Các toán tử có thể định nghĩa lại: ++, --,+, -, *, /, %, ^, &, !, ~, |, =, >, <, +=, -=, *=, /=, %=, ^=, &=, |=, <<, >>, ==, !=, <=, >=, &&, ||, [], (), new, new[], delete, delete[]
- Các toán tử không thể định nghĩa lại: ., ?:, sizeof(), ::, .*, …

Ép kiểu bằng toán tử

- Dùng để ép kiểu một đối tượng của lớp sang một kiểu dữ liệu khác.
- Ví dụ: trong lớp PhanSo:

operator int()

```
PhanSo::operator int()
{
   return TuSo/MauSo;
}
```

Toán tử gán

- Dùng để gán đối tượng
- Nếu toán tử gán không có, C++ tự động thêm toán tử gán mặc định:
  - Gán giá trị từng thuộc tính từ đối tượng nguồn vào đối tượng đích (member-wise copy).
  - Lưu ý: thuộc tính con trỏ sẽ được gán bằng → 2 đối tượng cùng trỏ vào 1 vùng nhớ
- → Lớp đối tượng có thuộc tính con trỏ, nên định nghĩa toán tử gán

Toán tử gán

- Toán tử gán của lớp PhanSo

```
PhanSo& PhanSo::operator=(const PhanSo& ps)
{
   this->TuSo = ps.TuSo;
   this->MauSo = ps.MauSo;
   return *this;
}
```

Toán tử gán

- Cài đặt toán tử gán:
  - Các thuộc tính có kiểu dữ liệu cơ sở: thực hiện gán
  - Các thuộc tính con trỏ: cân nhắc việc gán. Có thể phải giải phóng vùng nhớ cũ và cấp phát vùng nhớ mới
  - Lưu ý: kiểm tra việc đối tượng gán cho chính nó.

Toán tử gán

- Đối với đối tượng có thuộc tính được cấp phát động, nên cài đặt 3 hàm sau
  - Hàm khởi tạo sao chép
  - Toán tử gán
  - Hàm hủy

Luyện tập

- Thiết kế và cài đặt
  - Lớp DaThuc