1. Tìm câu sai:
   1. Khi cần thiết, trình biên dịch tự động gọi toán tử chuyển đổi kiểu để tạo ra một đối tượng tạm
   2. Trong C++, chỉ có thể đa năng hoá các toán tử có sẵn
   3. Các tham số mặc định không thể sử dụng với một toán tử đa năng hoá
   4. C++ cho phép tạo mới và đa năng hoá các toán tử
2. Các toán tử không thể đa năng hoá trong C++:
   1. . .\*
   2. :: ?: size of
   3. . .\* :: ?: size of
   4. . .\* :: ?:
3. Phát biểu nào sau đây là không đúng?
   1. Toán tử >> là toán tử thuộc lớp ostream
   2. istream và ostream là hai lớp dùng để nhập và xuất dữ liệu trong C++
   3. cout là một đối tượng của lớp ostream
   4. cin là một đối tượng của lớp istream
4. Phát biểu nào sau đây là **sai** trong đa năng hóa toán tử (operator overloading)
   1. Có thể định nghĩa thêm toán tử mới ngoài những toán tử đã có trong C++.
   2. Không thể thay đổi số lượng tham số (số ngôi) của các toán tử
   3. Độ ưu tiên của các toán tử không bị thay đổi.
   4. Không thể thay đổi kết quả thực hiện của các toán tử trên các kiểu dữ liệu cơ sở (built-in)
5. Để sử dụng được toán tử += cho đối tượng của class
   1. Ta phải đa năng hóa (overloading) toán tử +=
   2. Chỉ cần đa năng hóa hai toán tử + và =
   3. Chỉ cần đa năng hóa toán tử + vì toán tử = không cần đa năng hóa
   4. Không cần đa năng hóa toán tử += vì hệ thống có sẵn
6. Khi thực thi đoạn chương trình sau kết quả sẽ là (c21)

class Phanso

{

private: int tuso,mauso;

public:

Phanso(int t=0, int m=1)

{

tuso=t;

mauso=m;

cout<<"("<<tuso<<"/"<<mauso<<")"<<endl;

}

Phanso(const Phanso&x)

{

tuso= x.tuso;

mauso=x.mauso;

cout<<"("<<tuso<<"/"<<mauso<<")"<<endl;

}

const Phanso& operator++()

{

tuso += mauso;

return \*this;

}

};

void main()

{

Phanso x(1,2);

Phanso y=++x;

}

* 1. Kết quả in ra là (1/2)(3/2)
  2. Chương trình bị lỗi khi biên dịch
  3. Kết quả in ra là (1/2)(1/2)
  4. Kết quả in ra là (1/2)

1. Xét đoạn chương trình khai báo class Data như sau (c30)

class Data

{

private: int a;

public:

const Data& operator ++();

void Increment(){ a = a+1;}

};

Phần cài đặt toán tử ++ nào sau đâu là đúng:

* 1. const Date& Date::operator++( ) {

Increment();

return \*this;

}

* 1. const Date& Date::operator++() {

Date temp = \*this;

temp.Increment();

return this;

}

* 1. const Date& Date::operator++() {

Date temp = \*this;

Increment();

return temp;

}

* 1. const Date& Date::operator++() {

Increment();

Date temp = \*this;

return temp;

}

1. Giả sử có một lớp C với các đối tượng obj1, obj2 và obj3. Nếu biểu thức sau hoạt động đúng, overload operator – (chồng toán tử trừ) phải

C obj1, obj2, obj3;

obj3 = obj1 – obj2;

* 1. Có 2 tham số.
  2. Có giá trị trả về.
  3. Phải khai báo protected.
  4. Là hàm toàn cục bạn với lớp C friend

1. Nạp chồng (overloading) liên quan đến việc viết ra 2 hay nhiều hàm mà
   1. Cùng tên và khác kiểu trả về
   2. Khác tên nhưng cùng danh sách đối số
   3. Cùng tên nhưng khác danh sách đối số
   4. Cùng tên và cùng danh sách đối số
2. Toán tử nào không thể được overload
   1. Toán tử ++
   2. Toán tử ::
   3. Toán tử ()
   4. Toán tử ~