Instituto Superior de Engenharia de Coimbra
Programação II

Exame da época normal 4-12-2008

Nome:_____

Número_____

1 – Qual será a saída resultante da execução deste programa? (escolha uma hipótese)

```
class Base {
public:
       virtual void funcV(){ cout << "\n funcV de Base"; }
       void g(){ cout << "\n g de Base"; }
};
class Especial: public Base {
public:
       virtual void funcV(){ cout << "\n funcV de Especial"; }
       void g(){ cout << "\n g de Especial"; }
};
void main(){
       Base * x []= { new Base, new Especial };
       x[0]->funcV();
       x[1]->funcV();
       x[0]->g();
       x[1]->g();
}
```

a)
funcV de Base
funcV de Especial
g de Base
g de Base

b)
funcV de Base
funcV de Especial
g de Base
g de Especial

c)
funcV de Base
funcV de Base
g de Base
g de Especial

d)
funcV de Base
funcV de Base
g de Base
g de Base

2 – Qual será a saída resultante da execução deste programa? (escolha uma hipótese)

```
class Num {
public:
       Num(int nn = 0){ cout << "\n Num()"; }
       ~Num(){ cout << "\n ~Num()"; }
};
class Teste{
       Num numero;
public:
       Teste(){cout << "\n Teste()";}
       ~Teste(){cout << "\n ~Teste()";}
};
void main(){
       Teste a;
}
```

a)
Teste
~Teste

b)
Num
Teste
~Num
~Teste

c)
Num
Teste
~Teste
~Ñum

d) Dá erro porque o construtor de `Teste` não inicializa o membro `numero`.

3 – Qual será a saída resultante da execução deste programa? (escolha uma hipótese)

```
class Num {
	int n;
public:
	Num(int nn){ n = nn;}
	void setN(int nn){ n = nn;}
	int getN()const{ return n; }
};
class Teste{
	Num numero;
public:
	Teste():numero(0){}
	Num getNumero(){ return numero; }
	void alteraN(int nn){ numero.setN(nn);}
};
void main(){
	Teste a;
	a.getNumero().setN(2);
	cout << a.getNumero().getN() << endl;
	a.alteraN(4);
	cout << a.getNumero().getN() << endl;
}
```

a)

```
0
0
```

b)

```
2
4
```

c)

```
2
2
```

d)

```
0
4
```

4 - Considere as seguintes definições:

```
class Planta{
public:
	virtual void f()=0;
	virtual void g()=0;
};
class Arvore: public Planta{
public:
	virtual void f(){ cout << "f() de Arvore "; }
	virtual void h()=0;
};
class ArvoreDeFruto: public Arvore {
public:
	// . . .
};
```

Para que a classe `ArvoreDeFruto` seja concreta, isto é, para que possam existir objectos de `ArvoreDeFruto`, quais as funções que é preciso implementar nesta classe ?

a) É preciso definir as funções `g()` e `h()` na classe `ArvoreDeFruto` para que seja concreta.

b) É preciso definir as funções `f()`, `g()` e `h()` na classe `ArvoreDeFruto` para que seja concreta. Se faltar uma destas funções, a classe é abstracta.

c) Definindo apenas a função `h()` na classe `ArvoreDeFruto` esta fica concreta.

d) Para além de definir as funções funções `f()`, `g()` e `h()`, é ainda necessário definir outra função que não seja adquirida pior herança.

5 – Qual será a saída resultante da execução deste programa? (escolha uma hipótese)

```
class Base {
public:
	virtual ~Base(){ cout << "~Base \n"; }
};
class Deriv: public Base {
public:
	virtual ~Deriv(){ cout << "~Deriv \n"; }
};
void main(){
	Base * a = new Deriv;
	delete a;

	Deriv * x = new Deriv;
	delete x;
}
```

a)
~Base
~Deriv
~Base
~Deriv

b)
~Deriv
~Base
~Deriv
~Base

c)
~Base
~Deriv

d)
~Base
~Deriv
~Base

6 – Qual será a saída resultante da execução deste programa? (escolha uma hipótese)

```
void altera( int x, int &y, int *z) {
	++x; ++y; ++(*z);
	cout << "\n x=" << x << " y=" << y << " *z=" << *z;
}

void main() {
	int a = 1, b = 1, c = 1;
	altera( a, b, &c);
	cout << "\n a=" << a << " b=" << b << " c=" << c;
}
```

a)
x=2 y=2 *z=2
a=2 b=1 c=1

b)
x=2 y=2 *z=2
a=2 b=2 c=2

c)
x=2 y=2 *z=2
a=1 b=1 c=1

d)
x=2 y=2 *z=2
a=1 b=2 c=2

7 – Dado o seguinte programa:

```
class Num {
	int n;
public:
	Num(int nn){ n = nn;}
	void setN( int nn){ n = nn;}
	int getN(){ return n;  }
	// . . .
};
void main(){
	Num a(2), b(4);
	a << b;
}
```

Quais as funções que necessitam de estar definidas para que este código não tenha erros? (escolha uma hipótese)

a) `ostream & operator<<( ostream & saida, const Num & ob)` (função global)

b) `Num & operator <<( const Num & ob)` (função global)

c) `Num & Num::operator <<( const Num & ob)` (função membro da classe Num)

d) Não é preciso nenhum operador porque esta classe não tem ponteiros para memória dinâmica.

8 – Considere o seguinte programa:

```
double area( double comp, double larg=1) {
		return comp*larg;
}
double area() {
		return 10;
}
void main() {
	cout << "Area 1:" << area(4) << " Area 2: " << area() << endl;
}
```

Qual das seguintes afirmações é verdadeira ? (escolha uma hipótese)

a) Não é possível inicializar os parâmetros das funções no protótipo.

b) Não é possível ter duas funções diferentes com o mesmo nome.

c) Só os construtores das classes podem ter parâmetros com valores.

d) O programa corre bem, sem erros de compilação nem erros de execução.

9 – Dado o seguinte programa:

```
class Num {
	int n;
public:
	Num(){ n = 0; }
	void setN(int nn){ n = nn; }
};
void main() {
	Num x;
	const Num * p1 = &x;
	p1->setN(5);		// linha 3
	p1 = new Num();		// linha 4
	Num * const p2 = &x;
	p2->setN(5);		// linha 6
	p2 = new Num();		// linha 7
	const Num * const p3 = &x;
	p3->setN(5);		// linha 9
	p3 = new Num();		// linha 10
}
```

Indique quais as linhas deste programa em que ocorrem erros de compilação (escolha uma hipótese).

a) 4, 7, 10 **b)** 3, 7, 9, 10

c) 3, 6, 9 **d)** 3, 4, 6, 7, 9, 10

10 – Qual será a saída resultante da execução deste programa? (escolha uma hipótese):

```
class Teste{
	static int n;
public:
	Teste(){ ++n ; }
	static int getN(){ return n;}
};
int Teste::n = 0;

void main(){
	cout << "\nA- " << Teste::getN();
	Teste a[3];
	cout << "\nB- " << Teste::getN();
	Teste * p = new Teste[3];
	cout << "\nC- " << Teste::getN();
}
```

a)
```
0
1
2
```

b)
```
0
3
6
```

c)
```
0
3
3
```

d)
```
0
0
0
```