HERANÇA ENTRE CLASSES

Construção incremental de objetos:

- Composição: Objeto maior "contém", "é composto por" objeto menor.
- Herança: Objeto maior (também) é um objeto menor.
- Quando o objeto maior contém mais de um objeto menor, trata-se claramente de composição (e não de herança).
- Para que se caracterize herança, tudo que pode ser feito com um objeto menor também pode ser feito com um objeto maior.

Herança entre classes:

- Classe derivada herda (também possui) todos os membros da classe base, com exceções:
  - Construtores e destrutores
  - Operadores sobrecarregados
  - Funções amigas (friend)

Tipos de membros:

- public (todos têm acesso)
- protected (só os membros da classe e das classes derivadas têm acesso)
- private (só os membros da classe têm acesso)

Tipos de herança (quanto ao acesso):

- public (os public da classe base permanecem public)
- protected (os public da classe base se tornam protected)
- private (os public e os protected da classe base se tornam private)

Tipos de herança (quanto a de quem herda):

- simples: herda de uma única classe base
- múltipla: herda de mais de uma classe base

Substituição entre classes:

- Onde pode ser usado um objeto da classe base, tb pode ser usado um objeto da classe derivada.
- Onde se espera um objeto da classe derivada, não se pode usar um objeto da classe base.

```
Base B;
Deriv D;
Base  *pt1 = &D;  // OK
Deriv *pt2 = &B;  // Errado

void funcao(Base P1, Deriv P2);
funcao(B,D);   // OK
funcao(D,D);   // OK
funcao(B,B);   // Errado
```

Construtores e destrutores:

- Cada construtor da classe derivada deve indicar qual(is) dos construtores da(s) classe(s) base deve(m) ser chamado(s) quando ele for chamado

```
class Deriv: public Base
{ int P1;
public:
  Deriv( ...): Base( ... ), P1( ... ) { ... }
};
```

- Não existe equivalente no destrutor, pois só há um destrutor por classe.
- Ordem de execução:
  - Construtor: primeiro construtor(es) da(s) classe(s) base, depois construtor da classe derivada
  - Destrutor: primeiro destrutor da classe derivada, depois destrutor(es) da(s) classe(s) base

Sobscrevendo membros:
- Ao sobrescrever um membro da classe base na classe derivada, passará a ser usado normalmente o da classe derivada.
- Para acessar o membro da classe base que foi sobrescrito, é preciso usar o nome completo (classe::membro)