Herança e Manipulação de Arquivos

Um dos conceitos mais interessantes em programação orientada a objetos é o de herança. Trata-se de um mecanismo que permite que uma classe herde membros (atributos e/ou métodos) de outra classe. Isso permite aumentar reuso de código, produtividade e simplicidade na programação.

Na herança, membros comuns a diferentes classes são reunidos em uma única classe, conhecida como classe base ou superclasse. A partir da classe base, outras classes (chamadas classes derivadas ou subclasses) podem ser definidas possuindo os mesmos membros especificados na classe base. As classes derivadas podem conter membros que sejam particulares a elas, ou seja, não são compartilhados com as outras classes derivadas. A sintaxe para implementação de classes base e derivadas é bastante simples e intuitiva na linguagem C++. Veja nos slides abaixo um conjunto de exemplos. Atente também para o uso do modificador de acesso *protected*.

No contexto de herança, o conceito de polimorfismo agrega flexibilidade em programas complexos. C++ permite que um ponteiro (ou referência) para uma classe derivada seja tratado como um ponteiro para a classe base, um mecanismo conhecido como *upcasting*. É também possível que um ponteiro (ou referência) para a classe base seja convertido para um ponteiro para a classe derivada, um mecanismo conhecido como *downcasting*. Por fim, um método de uma classe base pode ser redefinido pelas suas classes derivadas, porém ainda pode ser acessado por um objeto da classe base utilizando-se métodos virtuais.

O mecanismo de herança também permite a definição de classes abstratas (não é possível instanciar objetos de classes abstratas), que servem de base para a definição de outras classes, e interfaces, que definem apenas o que o objeto expõe, sem se ater a detalhes de como realizar a implementação.

Para concluir o tema da semana, veremos a manipulação de arquivos em C++. Com os conceitos de herança e sobrecarga de operadores claros, a manipulação de arquivos torna-se bastante simples pela utilização de objetos *streams*. O conceito de *stream* pode ser entendido como um fluxo de informação que pode entrar ou sair de um programa para uma fonte de informação que pode ser um arquivo, a memória, o teclado ou o vídeo. Desta forma, a escrita para qualquer um destes meios utiliza basicamente a mesmas classes e métodos, facilitando seu uso.

Slides

- **Conjunto de slides do grupo de professores LP1 (material 2020.6)**
  - https://docs.google.com/presentation/d/1M2VQDoTn9ylT8M8mGySyO4TH4avfXO-zvgplSnq_UFA/edit?usp=sharing
  - https://docs.google.com/presentation/d/1_D5bx79aKeiEBN3KoN7snBNTnY4XmJnc_aSd3S1b4ac/edit?usp=sharing
- Exemplo destacando herança, classes abstratas, downcasting e upcasting: Link

Textos

- **Geek for Geeks (em ingles)**

    - Herança: Link
    - Arquivos: Link

- **Wikibooks - Programar em C++**
    - Herança: Link (em português)
    - Arquivos: Link (em inglês)

- **Livro C++ como programar**
    - Capítulos 12 e 13: Herança e Polimorfismo
    - Capítulo 17: Arquivos
    - DEITEL, Harvey M.; DEITEL, Paul J. C++ como programar. 5. ed. São Paulo: Pearson Prentice Hall, 2006. 1163 p. ISBN: 9788576050568

Vídeos

- **Aula sobre o tema referente aos slides do grupo de professores LP1 (material 2020.6)**
    - Herança: https://drive.google.com/file/d/1_nqQIXq_9hY1-b8Ig12htXK4nW322n1b/view?usp=sharing
    - Arquivos: https://drive.google.com/file/d/1vogk5-TJzToa-X5rZlOCueznrvQIgvak/view?usp=sharing

- **The cherno (Ingles, mas podem usar as legendas do Youtube)**
    - Herança: https://youtu.be/X8nYM8wdNRE (vejam também os videos de interface e métodos virtuais)