Abstract Classes

- An abstract class provides a common public interface for its class hierarchy, and are usually the base class.
- Class is made abstract by declaring 1 or more of its virtual functions to be "pure" in .h, no implementation in .cpp
  - virtual void print() const = 0;
  - Abstract classes are never instantiated (lack implementation)
- Derived classes must provide implementations for missing methods.
Abstract Classes

- Abstract class can have data members and concrete functions (constructors/destructors) which go by normal inheritance rules.
- Can use pointers to abstract classes to use polymorphic functionality on all concrete derived classes.
  - Useful with container classes (vector of abstract base class)
  - Can use iterator to iterate over items in container class
Function and class templates you specify with a single code segment an entire range of related (overloaded) functions or classes (function or class-template specializations).

Generic programming

Template specializations are tracings we make of stencils, same shape but maybe different colors.
Templates

- **Passing non template parameters**
  - `template< typename T, int elements > //compile time constant`
  - `Stack< double, 100 > smallstack;`
  - `.h could contain member: T stackHolder[ elements ];`

- **Type parameter can specify default type**
  - `template< typename T = string >`
  - `Stack<> jobDescriptions;`

- **Explicit specialization for a particular type**
  - `template<>`
  - `class Stack< Employee > { ... };`
Function Pointers

- A pointer to a function contains the address of the function in memory.
- Name of a function is actually starting address in memory of the code (like array name!)
- Function pointers can be
  - Passed to and returned from functions
  - Stored in arrays
  - Used to call the underlying function