OBJECT ORIENTED PROGRAMMING

Lab 4

OBJECT ORIENTED PROGRAMMING WITH C++

- Chapter Objectives
  - Arrays of objects
  - Using pointers to objects
  - Using new and delete
  - References

OBJECT ORIENTED PROGRAMMING WITH C++

- Arrays of objects
  - Objects are variables and have the same capabilities and attributes as any other type of variable.
  - The syntax for declaring an array of objects is exactly like that used to declare an array of any other type of variable.
  - Arrays of objects are accessed just like arrays of other types of variables.

OBJECT ORIENTED PROGRAMMING WITH C++

- Using pointers to objects
  - Objects can be accessed via pointers.
  - When a pointer to an object is used, the object's members are referenced using the arrow (->) operator instead of the dot (.) operator.
  - Pointer arithmetic using an object pointer is the same as it is for any other data type: it is performed relative to the type of the object.
  - For example, when an object pointer is incremented, it points to the next object.
  - When an object pointer is decremented, it points to the previous object.

OBJECT ORIENTED PROGRAMMING WITH C++

- Using **new** and **delete**
  - While **malloc()** and **free()** functions are available in C++, C++ provides a safer and more convenient way to allocate and free memory.
  - In C++, you can allocate memory using **new** and release it using **delete**. These operators take these general forms:

p_var = **new** type; // type can be a class or primitives

delete *p_var*; // p_var is a pointer for type

- Using **new** and **delete**
  - **new** is an operator that returns a pointer to dynamically allocated memory that is large enough to hold an object of type *type*.
  - **delete** releases that memory when it is no longer needed.
  - If there is insufficient available memory to fill an allocation request, one of two actions will occur.

    - return a null pointer // old way from C

    - generate an exception // Standart C++

OBJECT ORIENTED PROGRAMMING WITH C++

- References
  - A reference is an implicit pointer that for all intents and purposes acts like another name for a variable.
  - There are three ways that a reference can be used.
    - reference can be passed to a function
    - reference can be returned by a function
    - an independent reference can be created

OBJECT ORIENTED PROGRAMMING WITH C++

- References
- There are a number of restrictions that apply to all types of references.
- You cannot reference another reference.
- You cannot obtain the address of a reference.
- You cannot create arrays of references, and you cannot reference a bit-field.
- References must be initialized unless they are members of a class, are return values, or are function parameters.