Sunday, July 2, 2017

C++ Primer Plus

Compound Types

ARRAY

- Data form that hols several values of the same type
- Array element
- Size of the array can be constant expression
  - But all values need to be know at time compilation
- Compound type are build from other type
- First index is 0
- Compiler doesn't check valid subscript
- Sizeof operator
  - Return size in bytes
- Initialization rules for arrays
- Initialization rules for arrays
  - You can use the initialization form only when defining the array
  - You canvassing one array to another
  - When initializing you can provide fewer values than array elements
    - Compiler set remaining elements to 0
- Vector template class
  - Is an alternative to array from C++ Standard template class

STRING

- Strings
  - C-style string
    - Store string in an array of char with each character stored in its own array element

    - Special feature - last character of every string is a null character
    - Cout displays cat - it displays first four characters, detects null and stop
    - Cout displays dog - displays first five letters and then keeps marching through memory byte-by-byte, interpreting each byte as a character to print until null
  - String constant = string literal
    - Wasted space is no harm
    - C++ has no limit on the length of a string
      - When determining the minimum array size, include null character
    - Represents 2 characters - null and S
- Concatenating string constants
  - Any two string constants separated only by whitespace are joined into one
  - First character of second string immediately follows the last character from the first string
- Using strings in an array
  - #include <cstring> for strlen()
    - Get length of the string
- After null everything is ignored
- How cin determines when you have finished entering your string
  - we didn't get a chance to respond to dessert prompt
    - Program displayed it and moved on
    - Alistair is put to name array and Dreeb is sitting in queue
    - Dreeb is put to dessert question
  - You cant enter a null character from keyboard
    - So cin uses white spaces (spaces, tabs, newlines) to delineate a string
- How cin views string input
- Reading String Input a Line at a Time

- You need line oriented method instead of word oriented method
- cin.getline()
  - 2 arguments
    - name of array
    - Number of characters to be read
      - Limit is 20, function reads 19 at most (null)
  - It stops reading input when it reaches limit or end of the line
  - You can concatenate getline()
- cin.get()
  - It leaves new line character in the input queue
  - Second call sees n
  - Newline character and concludes that there is nothing to read
    - You can overcome it by calling cin.get() that will read the newline character and you can you get again
  - You can concatenate get()

- What happens after getline() and get() reads empty line
  - get()
    - After get() reads empty line, it sends back failbit
    - You can restore input with cin.clear()
  - getline()
    - The next statement picked up is where the last getline() left off
- What happens when the input string is longer than allocated space
  - Get() and getline() leave the ramaining characters in the input queue
  - However getline() additionally sets the failbit and turns off further input
- Mixing string and numeric input
  - When cin reads year, it leaves the newline int queue

- It can be fixed by many ways
- String class
  - Type string instead of array of chars
  - #include <string>
    - Part of the std namespace (std::string)
  - Main difference - you create string object as a variable, not an array
  - String operations
  - Uninitialized string object is set to zero
  - Reading a line into a string
    - No dot notation when you compare it with array
    - That indicates that this getline() is not a class method

STRUCTURE

- Structures
  - Can hold items of more than one data type
    - You can store info about person - name, age, grade
  - Structure declaration
    - struct - indicates that code defines the layout for a structure
    - Inflatable - name of the new type
  - Create variables
  - You can access its members
    - Hat is a structure, hat.volume is double
  - Where to place declaration
    - Inside main()
      - just after opening brace
    - External declaration
      - can be used by all functions following it

  - You can assign one structure to another using =
    - Member wise assignment
  - You can combine definition of structure with creation of structure variable
  - You can have structure without name and simultaneously define variable
- Arrays of structures
- Bit fields in Structures
  - Used in low-level programming

UNION

- Unions
  - Data format that can hold different data types but only one type at a time
  - You can use one4all variable to hold an int, long or double
  - Size of the union is the size of the largest member
- Anonymous union
  - Has no name

ENUMERATION

- Enumerators
  - For creating symbolic constants
  - It establishes symbolic constants (red, blue) for integer values 0-7
    - You can override it by your own values
    - Spectrum variable band is limited to just 8 values
  - Symbolic constats = enumerators
- Typecast
- Used for switch statement
- No enumeration type name
  - If you plan to use constants and not create variable of enumeration type

- Set values
  - First = 0, second = 100, this 101
  - All assigned values must be integers (long)
  - Zero = 0, null = 0, one = 1, numero_uno = 1
  - Enumerators can have same values
- Each enumeration has a ranch
  - You can assign any integer value in the range
    - Even if it isn't an enumerator value
  - Upper end range is defined as follow
    - When the greatest value is 101, then the smallest power of two is 128
    - So upper end range is 127
  - Lower end limit is defined as follow
    - Smallest value is 0 or greater -> lower limit range is 0
    - Smallest value is negative -> you the same approach as for upper limit

POINTERS AND THE FREE STORE

- Pointer - variables that store addresses of values rather than values themselves
- How to find addresses for ordinary variables
  - Home is variable, $Home is its address
  - Cout uses hexadecimal notation when displaying address values
- Pointers and the C++ philosophy
  - OOP emphasizes making decision during runtime instead of during compile time
    - Runtime
      - flexibility to adjust to current circumstances
      - Array is created in compile time
        - so C++ let you use keyword new to request the correct amount of memory and use pointer to keep track of where the memory is

- Treat the location as the named quantity and the value as a derived quantity
  - Pointer holds the address of a value
  - Name of the pointer represents the location
- * operator
  - = indirect value, dereferencing operator
  - Applying * yields the value at the location
- Variable and its pointer are just two sides of the same coin, completely equivalent
  - You can use pointer as you would use the variable
  - When you assign value to * p_myNumber it changes myNumber value
- Declaring and initializing pointers
  - Computer needs to keep track of the type of value to which a poster refers
    - Address of char looks the same as the address of a double but they use different number of bytes
    - p_updates points to type int
      - p_updates is a pointer, *p_updates is int
  - Int *ptr, int* ptr, int * ptr
  - int* p1, p2; //creates one pointer
    - Creates one pointer (p1) and one ordinary int(p2)
    - You need one * for each pointer variable name
  - Pointers (addresses) are the sane size, regardless of type they point to
    - Addresses are usually 2 or 4 bytes
  - Example
    - *pt and higgens are the same
    - &higgens and pt are the same
- Pointer danger

  - When you creates pointer computer allocates memory to hold the data to which address points

- Pointers and numbers
  - Pointers are distinct types from integers
    - You can't assign an integer to a pointer
    - Nothing tells the program that this number is an address
- Allocating memory with new
  - Pointers are especially useful when you locate unnamed memory during runtime to hold values
    - Pointers become the only access to that memory
  - In C (C++ too) you can allocate memory with malloc()
  - new operator
    - C++
    - You tell new what for what data type you want memory
      - New finds a block of the correct size and returns the address of the block
      - You assigns the address to the pointer
    - pn is address and *pn is value
      - Pointer points to a data object (not object as in OOP)
      - The memory to which pn points is not a variable
  - Computer might not have enough memory for new request
    - New returns value 0
      - Pointer with value 0 is called null pointer
      - C++ guarantees that the null pointer never points to valid data
- Freeing memory with delete
  - Return used memory to memory pool
  - It removes the memory to which ps points

    - It doesn't remove pointer ps itself
        - You can reuse ps to point to another new allocation
    - Memory leak
        - Memory that has. Been allocated but can no longer be used
    - Don’t free memory that you have previously freed
        - Result is undefined
    - You should delete only memory allocated with new
        - It is ok to apply delete to a null pointer
- Using new to create dynamic arrays
    - New is useful when you allocate larger chunks of data (arrays, strings, structures)
    - You are writing program that will maybe need an array
        - If you create program by declaring it, space is allocated when the program is compiled
            - Array is there, it doesn't matter if program will use it
        - Static binding
            - Allocating the array during compile time
                - Array is build in to the program at compile time
        - Dynamic biding
            - You can create an array during runtime if you need it and skip creating the array if you don't need it
                - Array is created while program is running = dynamic array
                    - Program can decide on away size while it is running
- Creating dynamic array with new
    - Array of 10 ints
        - New returns the address of the first element of the block
        - Address is assigned to pointer psome

    - Delete array after the use
        - Braces tell program to delete whole array, not just element pointed to by the pointer
    - Pointer points to first int
        - You have to keep track of how many elements are in the block
            - You can use sizeof operator to find it out
- Using a dynamic array
    - Effect of adding 1 to p3
        - The expression p3[0] now refers to the former second element of the array
        - Pointer now points to second element instead of first
    - example

POINTERS, ARRAYS AND POINTER ARITHMETIC

- Pointer arithmetic
    - Adding one to a pointer variable increases its value by the number of bytes of the type to which it points
    - 2 ways how to get the address of an array
    - Access elements with pointer notation
    - in many respects you can use pointer names and array names in the same way
        - As an address
        - Differences
            - you can change the value of a pointer, array name is constant
            - Sizeof
                - In this case sizeof doesn't interpret it as an address
- Pointers and strings
    - Cout assumes that address of a char is the address of string
        - So it prints out all characters in array until null

- Misuses that you should try to avoid
- C++ doesn't guarantee that string literals are stored uniquely
  - If use you string literal “wren” several times, compiler might store several copies or just one copy of wren
  - Normally if you give cout a pointer, it prints the address
    - If pointer us type char, it displays pointed-to-string
    - If you want address of string, you have to type cast the pointer to another type (int *)
    - Ps doesn't copy animal string, it copies the animal address
  - To get copy of the string
- strcpy() and strncpy()
  - It can cause problems if array isn't big enough
    - You can use stncpy()
      - It takes third argument - Max number to be copied

- Using new to create dynamic structures
  - Dynamic = memory is allocated during runtime
  - You can't use dot membership operator with the structure name
    - Because structure has no name
    - Use arrow membership operator (->)
  - Structure definer is
    - Name of the structure - use .
    - Pointer - use ->
  - Reusing freed memory
- Automatic storage, static storage, and dynamic storage
  - = free store = heap
  - Automatic storage

  - Ordinary variables = automatic variables
  - Exist automatically when function is invoke, expire when the function terminates
- Static storage
  - Exist throughout the execution of an entire program
  - Defined - outside of function = externally, use keyword static
- Dynamic storage
  - Pool of memory that is called a free storage
  - With new and delete you can allocate memory on one function and free it in another
  - Life of the data is not tied to program or function
- Memory leak
  - You can't access variable because pointer is gone and you can't delete it