Monday, October 26, 2015

Week Three

C++ starring Mr Feeney

1. array indices shortcut to offsets of memory
    1. ie x[5] is 5 units pass x[0]
    2. You have to be mindful to not let index pass the length of array
2. std::cont << “Address of a: ” << &a <<std::endl; //prints out address of the a[0]
    1. You can save address like this: int* address = &a; //where a is the variable
    2. You can’t just save an address to an integer, you have to save it into a “pointer”
        1. Pointer: int *p; //p is a pointer
        2. Pointer is an indirect reference to the variable
3. You can access the pointer’s pointed value by: std::cont << “Address of a: ” << *p << std::endl; //this is called “dereference"
4. You can pass parameter by:
    1. pass by value: the program makes a copy of the parameter, you can’t mess with it;
    2. pass by reference: the program passes the address of the variable, you can mess with it; //using &a
5. Fun Fact: in "x[0]", x is a pointer
6. Big lie: "strings" are dynamic arrays of "chars"
7. How to enlarge your array:
    1. DIRTY STREET TRICK: int* x = new int[array_size]; //this is a “dynamic array”, you can assign its length by a variable
        1. any issues?
            1. when you finish using the array, you have to: delete [] x;
            2. Otherwise the array with stick around;
            3. That’s why C/C++ are “unmanaged” languages
            4. You never know where the “end” of the array is
    2. C++’s USER FRIENDLY WAY: std::vector<int> vecX; //this is smart array!
        1. How big is dis? 0!
        2. How to stick stuff in there:
            1. vecX.push_back(8); //adds 8 into the vecX smart array
        3. vecX’s size:
            1. vecX.size(); //prints out the size of the smart array

8. more kinds loops:
    1. while (CONDITION TO CONTINUE) { //loop code } //while loop
    2. do{ //loop code } while(CONDITION TO CONTINUE) //do-while loop: does the code first, and then check for condition
9. How to make a function:
    1. RETURN_TYPE functionName(PARAMETER_TYPE PARAMETER…){ // code }
    2. RETURN_TYPE of “void": it means you don’t wanna return anything
10. Why should care about functions?
    1. You can then use "functionName(PARA);" to call the function, which exec the code in there!
    2. A team would easier to collaborate by working on different parts of the code at the same time
    3. Makes your code more elegant
11. {} are denoting stack! They are officially called scope. What happens in scopes stays in scopes.
12. Global in scope:
    1. *Dodge way*: declare your variable in front of all functions
13. Programming be like: “You code a lot, then you are back 10 minutes ago.” — Mr Feeney 2k15