PA2: Serialize

Due Date

- See Piazza for any changes to due date and time
    - Friday ***April 15*** before midnight
    - Grading the next day
- Submit all files and directories to Perforce
    - Create a directory called: PA2 in your student directory
    - Make sure you run ***CleanMe.bat*** before any submission
        - Failure to do so will harm your grade
    - /student/<yourname>/PA2/…
    - Fill out the submission report and add it to the directory

Goals

- Serialization
- Hands on programming, to serialize data structures

Assignments

1. Use the supplied project solutions given as a starting point.
    a. Your Work
        i. This assignment is designed to have minimal changes.
        ii. Classes , headers are provided
    b. You are not allowed to change the test code in any way!
    c. Majority of your work will be in:
        i. Serialization/ deserialization methods
        ii. Possible reordering / padding for the data structures in *.h
        iii. Fixing / adding of constructors
        iv. ***NO CHANGES*** in the destructors allowed
    d. ***All data*** needs to be initialized
        i. If you add padding or rearrange the data in a structure/class
            1. Make sure all data is initialized
        ii. Failure to do so will result in lost points
        iii. Do not trust on default constructors
            1. explicitly initialize every member

2. The compile the program and run.
    a. It contains 6 tests
        i. SampleClass
            1. It's complete so, you really have 5 tests to complete
                a. Test Fixture
                b. Serialization
                c. Unit Test function for the class
            2. Review this sample class
                a. It should help
        ii. Dog
            1. Test is provided
            2. Data structure clean up
                a. Reorder any data members in Dog.h
                b. Add the appropriate padding
                c. Correct the corresponding constructors (if needed)
            3. Add Serialization / Deserialization
            4. Test Code
        iii. Cat
            1. Test is provided
            2. Reduce packet size
                a. Find the original data size of the structure.
                b. Look at the data alignment
                c. Reorder the data to reduce size (smaller packets ☺)
            3. Data structure clean up
                a. Reorder any data members in Cat.h
                b. Add the appropriate padding
                c. Correct the corresponding constructors (if needed)
            4. Add Serialization / Deserialization
            5. Test Code
        iv. Bird
            1. Test is provided
            2. Data structure clean up
                a. Reorder any data members in Bird.h
                b. Add the appropriate padding
                c. Correct the corresponding constructors (if needed)
            3. Add Serialization / Deserialization
            4. Test code
            5. Make sure the destructor works correctly
                a. Put print statements to verify

v. Fish (see diagram below)
    1. Test is provided
    2. Data structure clean up
        a. Reorder any data members in Fish.h
            i. Including Fish, Orange, Apple structures
        b. Add the appropriate padding
        c. Correct the corresponding constructors (if needed)
    3. Add Serialization / Deserialization
    4. Test code
    5. Make sure the destructor works correctly
        a. Put print statements to verify
        b. Remember you cannot modify the destructor

vi. Medusa which holds many snakes (see diagram below)
    1. Test is provided
    2. Data structure clean up
        a. Reorder any data members in Snake.h
            i. Including Snake, SnakeLink, Egg, Medusa
        b. Add the appropriate padding
        c. Correct the corresponding constructors (if needed)
    3. Add Serialization / Deserialization
    4. Test code
    5. Make sure the destructor works correctly
        a. Put print statements to verify
        b. Remember you cannot modify the destructor

3. Write the necessary code to complete the above tasks
    a. You need to write any and all functions to make it work.
    b. Boost is ***NOT*** allowed!

Validation

Simple check list to make sure that everything is checked in correctly

- Program compiles and runs without crashing?
- Did you run the CleanMe.bat before any submit to Perforce.
- Program warning free?
- Is the data size of Cat smaller in size?
- Are the destructors releasing memory in Bird and Fish?
- Complete all classes to pass all tests?

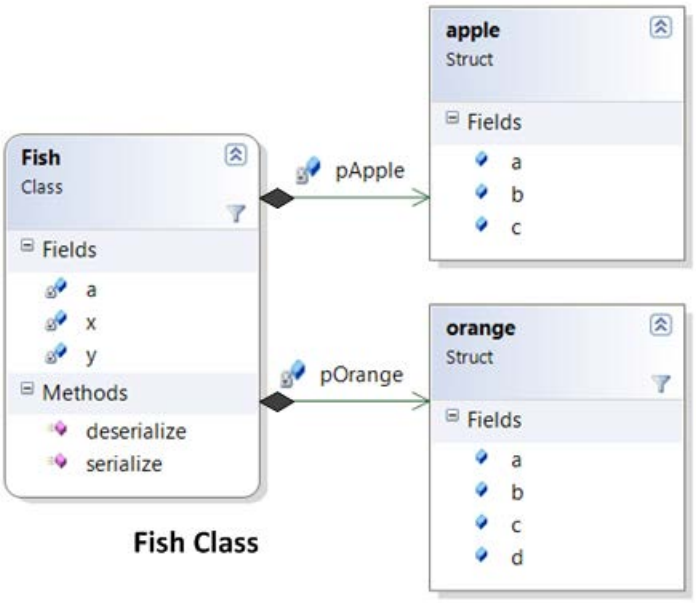
Hints

Most assignments will have hints in a section like this.

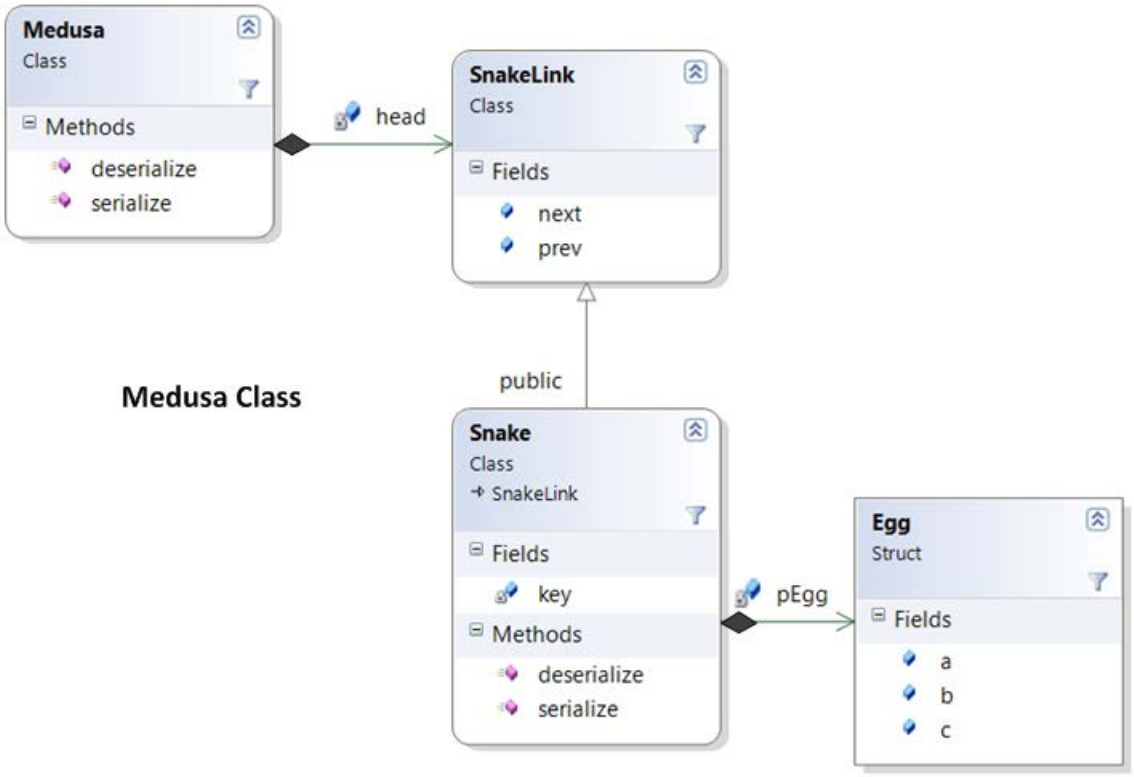
- Baby steps, use an very incremental process
  - Big steps will prevent you from finishing task
- I created an easy framework
  - Look at this code before the final exam
  - Similar problems will be on the final exam
- Review class notes

Troubleshooting

- Nothing special to report



Fish Class



Medusa Class

Question and Answers from previous classes:

Question: Fish

I assume we won't call new apple[] or new orange[] right?Just new apple() or new orange() right?

Answer:

1 apple and 1 orange are inside 1 fish.
You can create fish[], but that will create several fish, each with 1 apple and 1 orange at a time

---

Question: Bird

I am not really sure on how to begin the serialization/deserialization of char *s. I looked at the lecture, but I am not sure on how to tackle down this problem. Any ideas on where to start?

Answer:

So with serialization, you are basically packing your data into a single buffer so that you can store it or send it over the network. So with a char*, you can't just write out the pointer, because that memory address will change if you're sending the data somewhere or loading it in at a later time. So what you need to do is write the data that it's pointing to instead. There are several options here:

1. You're working with null terminated strings, so you can just put the string and the null terminator into the buffer. That way, when you're deserializing it, you know you're at the end of the string when you read in a null character.
2. You're working with raw binary data (which can contain null characters, i.e. an executable or an image). In this case, you'll want to serialize the length of the string followed by the string. When deserializing this data, you would first read in the length of the string, and then read that many bytes in after that.

The latter is more flexible as it can be used for both null terminated strings and raw data, but the former is what you're most likely used to dealing with.

Question: "Fixed size of string"?

In Bird, was it possible to have a String of dynamic length (i.e. the opposite of a fixed-size string). The constructor did not support passing in a string, and the length of 50 was hardcoded into it. The assignment never said Bird needed to support a dynamic string:

iv. Bird
1. Test is provided
2. Data structure clean up
   a. Reorder any data members in Bird.h
   b. Add the appropriate padding
   c. Correct the corresponding constructors (if needed)
3. Add Serialization / Deserialization
4. Test code
5. Make sure the destructor works correctly

Answer:

As I stated in class, I pushed fixed data to allow unit tests to work.
Your code should serialize the class,

Strnlen() is your friend,

don't hard code,
don't hard code,
don't hard code!

Question: Copying Issues

I know this assignment is supposed to be easy... but I'm having a bit of trouble with one part. The deconstructor for Bird is bombing and I believe its because I haven't correctly made a new version of s for it to delete on the newdata. However I can't figure out how to make that copy in the serialize function. I know it should be easy.. but I can't seem to figure it out.

Answer:

You should only be making a new copy in the Deserialize function. When you're reading in your string, you have to get the length of the string from the buffer (either because you added the length or by reading until you hit a null terminator -- it depends how you implemented your serialize). Then you would do:

s = new char[length + 1]; (The + 1 because I am assuming your length didn't include the null terminator. If it did, then ignore the + 1).

Question: Alignment fixing

I think I already know the answer to this, but are we allowed to use use single byte alignment packing for classes for this assignment or the rest of the quarter? The reason I'm asking is because there is a definite serialization speed increase for objects that don't contain pointers if the byte alignment is set to 1 because we can just serialize the object directly into the buffer without any extra padding being there.

Example of what I mean:

```
#pragma pack(push)
#pragma pack(1)
class myClass
{
   int x;
   char y;
   int z;
};
#pragma pack(pop)
```

I mainly would like to use this so that in specific cases where I know I am not dealing with pointers, serialization/deserialization becomes a little easier.

Answer:

No that is illegal in class <period>
Any additional pragmas or language tricks gives you the grade of 0
