Bernardo Fitzmaurice Acevedo 105297603

Class code: COS30031

Start date: 02/08/2024

Task 02 Lab C++ for Programmers

* Q.1 What is the difference between a struct and a class?
  + Structs is a group of values, while class is a type with a value defined by the user.
* Q.2 What are function declarations?
  + Bind a function to a name, and I give it a return type and parameters.
* Q2.1 Why and when are they needed?
  + They are needed so that the function can be used throughout the code, and they are used when you create a function that you want to use in more than one part of the code.
* Q.3 Why are variable names not needed here?
  + Because the only the function’s interface is specified.
* Q3.1 Could you add variable names? Would that be good?
  + Yes, I could add more variables. I don’t think it would be good, because it’s not necessary.
* Q.4 Does your IDE know if this method is used? If yes - how does it indicate this? (Colour? Tip? Other?)
  + Yes, it can know if it’s used. It grays out the color of the function if it’s not used.

**Section 1**

* Q.5 un-initialised values ... what prints and why?
  + It prints age, x and y equal 0. It just prints garbage values because the values are indeterminate.
* Q.6 Did this work as expected?
  + Yes.
* Q.7 Initialisation list - what are they?
  + A technique to initialize an object’s data members in a constructor.
* Q.7.1 Does your IDE suggest what the values are?
  + 0 for all, age, x and y.

**Section 2**

* Q.8 Should show age=1, x=1, y=2. Does it?
  + No, age is 1, but x is 2 and y is 3.
* Q.9 Something odd here. What and why?
  + The age is odd, it marks age as 4294967295.

**Section 3**

* Q.10 showParticle(p1) doesn't show 5,6,7 ... Why?
  + Because p1 is set as {1,1,1}

**Section 4**

* Q.11 What does -> mean?
  + Is an access operator.
* Q.12 Do we need to put ( ) around \*p1\_ptr?
  + Yes. Because the dot ‘ . ’ operator has higher precedence than the ‘ \* ’ operator.
* Q.13 What is a dereferenced pointer?
  + Is the actual value that the pointer points to. It’s used with ‘ \* ’.
* Q.14 Is p1 stored on the heap or stack?
  + p1 is stored the stack.
* Q.15 What is p1\_ptr pointing to now? (Has it changed?)
  + Is pointing to “createParticleWith(7, 7, 7)”. It has not changed.
* Q.16 Is the current value of p1\_ptr good or bad? Explain
  + Is bad. Because it points to the old memory location of ‘p1’. That is no longer relevant.

**Section 5**

* Q.17 Uncomment the next code line - will it compile?
  + It gives an error message, and I can keep compiling if I want, but the line will do nothing.
* Q.18 Does your IDE tell you of any issues? If so, how?
  + It gives me 2 error messages: “Unexpected tokens…” and “’getPrticleWith’ identifier not found”
* Q.19 MAGIC NUMBER?! What is it? Is it bad? Explain!
  + Is a numeric literal that is used in the code without explanation. It is bad, because it reduces the flexibility of the code and increases the chance of errors.
* Q.20 Explain in your own words how the array size is calculated.
  + We can calculate the size of the array if we divide the total size of the array by the single of a single element (a.k.a. ‘sizeof(p\_array1)’ / ‘sizeof(p\_array1[0])’).
* Q.23 Change the size argument to 10 (or similar). What happens?
  + It tries give answers for age, x and y that are out-of-bound, so it ends ou as garbage values.
* Q23.1 You might see some values that we set earlier. Why would this happen?
  + Because it will access all the stored date from previews variables or operations first.

**Section 6**

* Q24 Points to nothing - does it?
  + Indeed, it pointed to nothing, but I added and assigned ‘new Particle()’ the pointer starts pointing to a new memory space.
* Q.25 What is "hex" and what does it do? (url in your notes)
  + Is a method of stream manipulation in C++. It sets the baseline for Hexadecimal numbers. (<https://www.geeksforgeeks.org/ios-manipulators-hex-function-in-c/>)
* Q.26 What is new and what did it do?
  + Is an operator used to allocate memory dynamically for an object or a variable during the execution of the program. It saved a new ‘Particle during the execution of the program’.
* Q.27 What is delete and what did it do?
  + Is an operator used to deallocate (remove) memory that was previously allocated using the operator ‘new’. It deleted the memory previously saved by ‘new Particle()’.
* Q.28 What happens when we try this? Explain.
  + It checks for ‘nullptr’ before dereferencing to prevent undefined behavior. But I modified it so that if ‘p1\_ptr != nullptr’ it evaluates to ‘false’ so the program does not crash.
* Q.29 What is the difference between NULL and nullptr and 0?
  + Null is a macro that represents the value 0, is used to represent a null pointer.
  + ‘Nullptr’ is used as a type-safe way to represent a null pointer. Is better to use than ‘null’ because is only for pointers.
  + 0 is a number and can also represent a null pointer. Is less preferable than the other two, since it can cause more errors in the code.
* Q.30 What happens in this line? (A zero address now, so ...)
  + The pointer is ‘nullptr’ after the pointer was deleted and is evaluated as null.

**Section 7**

* Q.31 Are default pointer values in an array safe? Explain.
  + Yes, because they are pointing to ‘nullptr’ and don’t crash unless I move something in the code.
* Q.32 We should always have "delete" to match each "new". What is the problem if we don't delete, and what is the common name for this?
  + If we don’t ‘delete’ we can cause a memory leak, and can exhaust the system’s memory resources.
* Q.33 Your IDE may have tools to help you track memory. Does it?
  + Yes, it has.
* Q.34 Can you see what happens if you DON'T do this?
  + Yes, if the allocated memory is not deleted, after running the program multiple times the memory used will not be free, consuming more system memory.
* Q.35 Should we set pointers to nullptr? Why?
  + Yes, to prevent the pointers from calling garbage data or even crashing the code or data corruption.
* Q.36 How do you create an array with new and set the size?
  + ‘Particle\* ptr\_array = new Particle[size];’. After using the array it’ll be good to use a ‘delete[ ] ptr\_array;’ to avoid any problems with the pointer.

**Out of the main**

Q.21-22 Go to the showParticleArray 2 implementation and see there:

* Q.21 What is the difference between this function signature and the function signature for showParticleArray?
  + The difference is how they pass the array. The function signature receives a pointer to the first element of the array. While the function signature for showParticleArray receives the array as a parameter.
* Q.22 Uncomment the following. It gives different values to those we saw before so it won't work as a way to determine array size - but why?
  + Because ‘sizeof(arr)’ is passing as a function parameter, so it’s returning the size of the pointer and not the entire array.