Lab 2.p Part 2

##Q.1 What is the difference between a struct and a class?
a class is a container for methods and data
a struct is a data type you make that contains multiple datatypes, example is a Vector3D has x, y, z set as integers

##Q.2 What are function declarations?
Function declarations are a 'setup' for a function, it declares its name and type with no output or functionality
// ##Q2.1 Why and when are they are needed.
they are needed in case of dependency, in a large project you may need to call upon a function but it might not have been declared yet, if you declare it you can then use it in other parts of your project.

##Q.3 Why are variable names not needed here?
for this function names are not needed as we are just declaring data types and not using them.
##Q3.1 Could you add variable names? Would that be good?
its possible however you are just adding names with no purpose, since this is just a declaration your not going to use them.

##Q.4 Does your IDE know if this method is used?
Yes, this is shown by a green underline if its not been used and informs that the function is not being used, the same for variables (this is for VS)

##Q.5 un-initialised values ... what prints and why?
It will error out, since there is nothing there to print a compile error will occur.

<< Section 1 >>
Q.5: a with uninitialised values ... Particle: (age=0), (x,y)=(0,0)
Q.6: a with assigned values 0,10,20 ... Particle: (age=0), (x,y)=(10,20)
Q.7: b with initialised values 0,0,0 ... Particle: (age=0), (x,y)=(0,0)

##Q.6 Did this work as expected?
Yes, everything in the cout statements printed out the expected output (seen above)

##Q.7 Initialisation list - what are they?
Initialisation lists are used to call a constructor for an object that takes arguments with data types from an object of another class.

##Q.7.1 Does your IDE suggest what the values are?
No

##Q.8 Should show age=1, x=1, y=2. Does it?
Not exactly, the actual output is
“Q.8: p1 with 1,2,3 ... Particle: (age=1), (x,y)=(2,3)”
So the data is the same but the way its formatted is different.

##Q.9 Something odd here. What and why?
The input for age is -1 but the output is as followed
“Q.9: p1 with -1,2,3 ... Particle: (age=4294967295), (x,y)=(2,3)”
This is due to age being a unassigned int. making it not being able to set a - or + value, always being a non negative number. Setting to -1 will set the address in memory to all 1’s thus showing the 429…, this can be fixed by removing the unassigned keyword, resulting in:
“Q.9: p1 with -1,2,3 ... Particle: (age=-1), (x,y)=(2,3)”

##Q.10 showParticle(p1) doesn't show 5,6,7 ... Why?
In this section we use a function “setParticleWith”, the original code didn’t use a pointer so when we “changed” the values in p1 we actually made a copy of p1, change its values then tried to print the original instance. Making the argument for particle to a pointer allowed the “setParticleWith” edit the value using the stored memory address, thus directly editing the object.

##Q.11 What does -> mean?
The -> symbol is used to access members of a class, struct or array that are making use of pointer variable type.
So for Class* Player, to access the variable Location you need to use the syntax Player->Location

##Q.12 Do we need to put ( ) around *p1_ptr?
Yes, since we are using * (dereference) we are using the pointer as an actual value of what the pointer is pointing to rather then the memory address, however an alternative is just using p1_ptr->age as the -> does the task of dereferencing and getting using the . operator.

##Q.13 What is a dereferenced pointer?
A dereferenced pointer is a pointer that you use the * operator on to access or manipulate the data on what the pointer was pointed to. Example
int* pX = &x; pointer to variable x
*pX ++; * is making the pointer into a dereferenced pointer.

##Q.14 Is p1 stored on the heap or stack?
P1 is on the stack, heap is reserved for things that are declared with the NEW (and other) key words

##Q.15 What is p1_ptr pointing to now? (Has it changed?)
Nothing has changed, still pointing to p1

##Q.16 Is the current value of p1_ptr good or bad? Explain
Good, looking at the values in the debugger its still set to the address of p1 and showing the same data as p1 thus making it good.

##Q.17 Uncomment the next code line - will it compile?
No, since there is no method declared with the name “getParticleWith”

##Q.18 Does your IDE tell you of any issues? If so, how?
Ide tells me that “getParticleWith” is undefined, this is done by checking for declarations for the method in any code that is executed before the line.

##Q.19 MAGIC NUMBER?! What is it? Is it bad? Explain!
Assuming that the magic number referenced here is the 3 in showParticleArray(p_array1, 3);. It’s the input for size, being used for the for loop to print out each array element. This is bad as its not dynamic, if you make an array with 4 elements it wont function as intended. Solution would be to do a for each, looks something like for (int I = 0; p_array), this will loop for amount of elements in the array, making this method more dynamic.

##Q.20 Explain in your own words how the array size is calculated.
Its calculated by getting the total amount of elements (of every array element) and dividing it by the amount of a single element’s element count, thus getting the amount of particles in p_array1 (36 total elements / 12 (amount in 1) = 3)

##Q.21 What is the difference between this function signature and the function signature for showParticleArray?
the first method pass a dereferenced pointer and the second only passes the pointer array

##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 keyword needs an array for input, not a pointer. Its trying to get the size of a memory address and not an array.

##Q.23 Change the size argument to 10 (or similar). What happens?

##Q23.1 You might see some values that we set earlier. Why would this happen?

You get -858993460 for all uninitialized elements in the array, this is the default value for an uninitialized int being printed out done by the compiler for debugging reasons.

// ##Q24 Points to nothing - does it?

It points to nothing and errors out “uninitialized local variable”, however the error only occurs when trying to use the pointer, not when setting it.

##Q.25 What is "hex" and what does it do? (url in your notes)

Hex refers to hexadecimal and it’s a i/o manipulator in the context of the cout statemen, making the next value into a hexadecimal value. Printing out the memory location of p1 in hexadecimal.

##Q.26 What is new and what did it do?

New keyword is used to assign memory for objects and other data types dynamically and assigning it to the heap. However you need to delete the memory after use otherwise you could potentially have a memory leak.

##Q.27 What is delete and what did it do?

Delete removes data from the heap.

In this code we used new to create a particle and assigned it to the heap. After we finished using it we needed to remove it from the heap so we use the delete keyword.

##Q.28 What happens when we try this? Explain.

The pointer will appear as undefined, showing us a default undefined value. Also in VS, it will show values within the object as “unable to read memory”

##Q.29 What is the difference between NULL and nullptr and 0?

NULL is just a macro for a int 0, they are technically equal to each other, nullptr however is its own value used to represent a pointer address set to 0.

##Q.30 What happens in this line? (A zero address now, so ...)

P1_ptr will appear to be equal to int 0, or when printed it will show a memory address of 0x0000…

##Q.31 Are default pointer values in an array safe? Explain.

If not set to null pointer or deleted, they can be considered unsafe as they may end up causing memory leaks. It depends on how they are used or initialized.

##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?

We can have a memory leak if we don’t every delete each new variable.

##Q.33 Your IDE may have tools to help you track memory. Does it?

VS shows us active memory in usage, as well as what variables are being used and their memory addresses. We can also add a watch to variable to see what a specific variable does during runtime.

##Q.34 Can you see what happens if you DON'T do this?

Nothing shows up in my IDE, I assumed the pointer values would still show up on memory usage.

##Q.35 Should we set pointers to nullptr? Why?

Yes, in the case we end up trying to re use the deleted pointer we wont get an exception error
https://cplusplus.com/forum/general/282862/ (needed to look this one up)

##Q.36 How do you create an array with new and set the size?

If we have an array (arr), after initializing it with the new keyword we would add int[size] after to set the size: int *arr = new int[4];