iClicker

C++ vs Java:

A. C++ requires compiling on the target system
B. Java is ported by copying the “byte code” of an application
C. C++ applications can be run on another system if the processor and OS match
D. All of the above
E. B and C

C++ header files:

A. Can contain any valid C++ source
B. Must contain function declarations
C. must accompany a corresponding source file
D. Name much match the corresponding CPP file
E. Must end with “.h”

C++ Parameter Passing:

A. Call-by-value
B. Call-by-reference
C. Call-by-value unless otherwise specified
D. Call-by-reference unless otherwise specified
E. I didn’t really get this on Tuesday

C++ Functions:

A. Are identical in every way to Java methods
B. Can not be included in a class
C. Must be included in a class
D. May be in a class or outside of a class
E. I don’t know

Data Types

Example C++ Data Primitives:

- int
- unsigned int
- char
- unsigned char
- float
- bool

https://www.tutorialspoint.com/cplusplus/cpp_data_types.htm

Let's talk about integers.

Declaration with no initial value:

```
char              cStudentsInLecture;
int               iStudentsInLecture;
unsigned char     ucStudentsInLecture;
unsigned int      uStudentsInLecture;
unsigned short    usStudentsInLecture;
unsigned long     ulStudentsInLecture;
```

Declaration with Initialization:

```
unsigned short studentsInLecture = 118;
```

Note: any code I show you that does not conform to standards expected in code review does not give you permission to emulate that "style"

Note: the prefixes "i", "u", "us", "ul" are used here just to distinguish these declarations from one another. This is known as Hungarian Notation and is not used in this course.

Remember:

- "Char" 8-bit
- "short" 16-bit
- "int" 32-bit
- "long" 64-bit
- In most instances, you do not need to make a distinction.
- There are disadvantages to using "short" and "char" to save space.

Pointers

1. are integers that are used to refer to locations in memory.
2. can specify what data type is expected to be at that memory location. This includes user-defined data or even executable logic. This impacts how that data is treated when we access it.
3. can be declared with a NULL value but must be initialized to be used.

Example declarations of pointer to an integer:

```
int * studentCountPtr;
int*  studentCountPtr;
int  *studentCountPtr;
```

Discussion: where should we place “*” (asterix)?

You can declare a pointer with an initial value as long as there is something already defined:

```
int totalStudentCount = 337;
int* studentCountPtr = &totalStudentCount;
```

OR we can initialize the value like this:

```
studentCountPtr = &studentsInLecture;
```

In this context, “&” means “use the address of”.

We access the value at studentCountPtr like this:

```
studentsInLecture = *studentCountPtr;
```

In this context, “*” means “use the value located at”.

References

References are quite a bit like pointers except:

1. There is no such thing as a NULL reference.
2. You can not declare a reference that has no value.
3. Pointers can change, references can not.

Declaring a reference to an integer:

```
int totalCS126StudentsAsleep;
int& sleepingCS126Students = totalCS126StudentsAsleep;
```

Note: just as with “*” when using pointers, the placement of “&” should also be carefully considered.

References and pointers:

- save the need to pass all values to a function
- Allow us to access or modify objects or data directly

Pointers can

- have a NULL value
- remain uninitialized until needed
- change value

References

- Can not have a NULL value
- Must be initialized when declared
- Can not change values

```
=======================================
Java: “HelloWorld.java”
=======================================
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}

=======================================
ANSI C: “Hello_World.c”
=======================================
#include <stdio.h>
main( ) {
    printf("hello, world\n");
}

=======================================
C++: “Hello_World.cpp”
=======================================
#include <iostream>

using std::cout;
using std::endl;

int main() {
    cout << "hello, world" << endl;
    return 0;
}
```

iClicker (Again)

The “&” operator

A. Refers to an address of something
B. Indicates a reference to something
C. Is context sensitive

The “*” (asterix) operator

A. Refers to the contents of a location indicated by a pointer
B. Declares a variable to be a pointer
C. Is context sensitive