SEP105 – Allowed Variable Types and Functions for use in Assignments

This is a list of the allowed functions, variable types, etc that we have covered in class to the end of Projetc 3. If a variable type, function (that you did not create), library (that you did not create), class (that you did not create), etc is not on this list, then it cannot be used in the assessment! If you think we have covered something on the unit site and it does not appear on this list, please contact the unit chair for clarification before using it in your code.

Week 2:

- #include <iostream>
- #include <string>
- int main()
- #define
- #define _CRT_SECURE_NO_WARNINGS
- Comments – both inline (//) and block (/* */)
- Define an integer variable type - int
- Define a long variable type – long or long long
- Define a floating-point variable type – float
- Define a double precision floating point variable type – double
- Define a Boolean type variable - bool
- Define a character variable type - char
- Define a character array/C-string variable type - char []
- Define a string variable type - std::string
- Define an unsigned variable type of specific byte length - uint8_t, uint16_t, uint32_t
- Define a signed variable type of specific byte length - int8_t, int16_t, int32_t
- Define a variable as unsigned - unsigned
- Define a variable as signed - signed
- Define a variable as constant (ie cannot change) - const
- Define a new type – typedef
- Retrieving the maximum value from a variable – CHAR_MAX, INT_MAX, UINT_MAX, LONG_MAX, ULONG_MAX
- Retrieving the minimum value from a variable – CHAR_MIN, INT_MIN, LONG_MIN
- printf()
- std::cout
- Streaming out - <<
- scanf()
- scanf_s()
- std::cin
- Streaming into - >>
- std::cin.getline()
- std::cin.get()
- getchar()

Week 2 – Only from the Examples:

- strcpy()
- strcpy_s()
- std::string.copy() – used within an object (eg std::string cat; cat.copy())
- std::string.c_str() – convert a string object to a c-string (eg std::string frog; frog.c_str())

Week 3 – From the classes:

- Comments – header blocks for all code
- Comments – header block for any functions
- Comments – License information
- Addition - +
- Subtraction - -
- Multiplication - *
- Division - /
- Brackets – ()
- Modulus - %
- Casting – for example, to cast a float as an int: float a; (int) a
- #include <cmath>
- Square Root – sqrt()
- Power – pow()
- Sin – sin()
- Cosine – cos()
- Tan – tan()
- Inverse Sin ($\sin^{-1}$()) – asin()
- Inverse Cosine ($\cos^{-1}$()) – acos()
- Inverse Tan ($\tan^{-1}$()) – atan()
- Inverse Tan all 4 quadrants ($\tan^{-1}$()) – atan2()
- Natural Logarithmic – log()
- Exponential – exp()
- Logarithmic base 10 – log10()
- Logarithmic base 2 – log2()
- Increment after a variable has been used - ++ (eg a++)
- Increment before a variable is used - ++ (eg ++a)
- Decrement after a variable has been used - -- (eg a--)
- Decrement before a variable is used - -- (eg --a)
- Shorthand Addition - +=
- Shorthand Subtraction - -=
- Shorthand Multiplication - *=
- Shorthand Division - /=
- Shorthand Modulus - %=
- Equal to - ==
- Not Equal to - !=
- Greater than - >
- Greater than or equal to - >=
- Less than - <

- Less than or equal to - <=
- Logical AND - &&
- Logical OR - ||
- Logical NOT - !
- if
- else
- else if
- Condition Operator - ?
- switch
- case
- break
- default

Week 4 – From the classes:

- while
- do while
- for
- continue
- break
- built in arrays 1D – int a[], double b[], etc
- built in arrays nD – int a[][]; (2D array), double b[][][] (3D array), etc
- #include <string.h>
- memset()
- sizeof
- struct
- accessing a component of a struct (eg a variable) – eg a.c, a.k(), etc
- #include <array>
- std::array<,>
- std::array.begin() – used within an object (eg std::array<int,3> dog; dog.begin())
- std::array.end() – used within an object (eg std::array<int,3> dog; dog.end())
- std::array.size() – used within an object (eg std::array<int,3> dog; dog.size())
- std::array.empty() – used within an object (eg std::array<int,3> dog; dog.empty())
- std::array.fill() – used within an object (eg std::array<int,3> dog; dog.fill())
- Multidimensional array – eg std::array<std::array<,>,>
- #include <vector>
- std::vector<>
- std::vector.push_back() – used within an object (eg std::vector<double> frog; frog.push_back())
- std::vector.insert() – used within an object (eg std::vector<double> frog; frog.insert())
- std::vector.pop_back() – used within an object (eg std::vector<double> frog; frog.pop_back())
- std::vector.erase() – used within an object (eg std::vector<double> frog; frog.erase())
- std::vector.resize() – used within an object (eg std::vector<double> frog; frog.resize())
- std::vector.begin() – used within an object (eg std::vector<double> frog; frog.begin())
- std::vector.end() – used within an object (eg std::vector<double> frog; frog.end())

- std::vector.size() – used within an object (eg std::vector<double> frog; frog.size())
- std::vector.empty() – used within an object (eg std::vector<double> frog; frog.empty())
- std::vector.shrink_to_fit() – used within an object (eg std::vector<double> frog; frog.shrink_to_fit())
- std::vector.back() – used within an object (eg std::vector<double> frog; frog.back())

Week 4 – Only from the Examples:

- std::array.front () – used within an object (eg std::array<int,3> dog; dog.front())
- std::array.back() – used within an object (eg std::array<int,3> dog; dog.back())
- std::array.swap() – used within an object (eg std::array<int,3> dog; dog.swap())
- std::vector.clear() – used within an object (eg std::vector<double> frog; frog.clear())
- NULL
- std::string.length() – used within an object (eg std::string cat; cat.length())

Week 5 – From the classes:

- Indirection operator (pointers) - *
- nullptr
- Address Operator (pointers) - &
- Double Pointers - **
- Functions – return types, arguments/parameters, prototypes, default arguments, overloading, pass by copy, pass by reference, pass by reference with pointers
- return
- Functions – recursion, tail recursion

Week 6 – From the classes:

- class
- public
- private
- protected
- accessing a components of a class (eg function or variable) – . eg a.func()
- Creating functions outside of a class definition – eg className::func()
- Class constructors
- explicit
- Class Destructor
- Pointers with a class – eg pointing to a class, accessing components of a pointed to class (eg a->b)
- Class public inheritance

Week 7 – From the classes:

- #include <fstream>

- #include <stdio.h>
- File Pointer – FILE *filePtr
- fopen()
- fputc()
- fputs()
- fprintf()
- fgetc()
- fegts()
- fscanf()
- fclose()
- std::fstream
- std::fstream.open() - used within an object (eg std::fstream file; file.open())
- std::fstream.write() - used within an object (eg std::fstream file; file.write())
- std::fstream.put() - used within an object (eg std::fstream file; file.put())
- std::fstream.peek() - used within an object (eg std::fstream file; file.peek())
- std::fstream.get() - used within an object (eg std::fstream file; file.get())
- std::fstream.getline() - used within an object (eg std::fstream file; file.getline())
- std::fstream.close() - used within an object (eg std::fstream file; file.close())
- std::ios::app
- std::ios::ate
- std::ios::in
- std::ios::out
- std::ios::trunc
- std::ios::binary
- End Of File character - EOF

Week 7 – Only from the Examples:

- Generate a Pseudo Random number - rand()
- Set the seed for a random number generator - srand()
- Return the current processor ticks used by the program - clock()

Week 8 – From the classes:

- #ifndef
- #endif
- #pragma once
- extern
- Including local files – eg #include “stringToNum.h”
- namespace
- using namespace

Week 9 – From the classes:

- #include <list>
- std::list<>

- std::list<>::iterator
- Go to the next node in the list – iterator++ - assuming iterator is of type std::list<>::iterator
- Go to the previous node in the list – iterator-- - assuming iterator is of type std::list<>::iterator
- std::list.begin() - - used within an object (eg std::list<int> log; log.begin())
- std::list.end() - - used within an object (eg std::list<int> log; log.end())
- std::list.push_back() - - used within an object (eg std::list<int> log; log.push_back())
- std::list.push_front() - - used within an object (eg std::list<int> log; log.push_front())
- std::list.insert() - - used within an object (eg std::list<int> log; log.insert())
- std::list.pop_back() - - used within an object (eg std::list<int> log; log.pop_back())
- std::list.pop_front() - - used within an object (eg std::list<int> log; log.pop_front())
- std::list.erase() - - used within an object (eg std::list<int> log; log.erase())
- std::list.remove() - - used within an object (eg std::list<int> log; log.remove())
- std::list.clear() - - used within an object (eg std::list<int> log; log.clear())
- std::list.empty() - - used within an object (eg std::list<int> log; log.empty())
- std::list.size() - - used within an object (eg std::list<int> log; log.size())
- std::list.front() - - used within an object (eg std::list<int> log; log.front())
- std::list.back() - - used within an object (eg std::list<int> log; log.back())

Week 10 – From the classes:

- Bitwise AND - &
- Bitwise OR - |
- Bitwise XOR - ^
- Bitwise NOT - ~
- Bit shift LEFT - <<
- Bit shift Right - >>