

As minimal preparation for the written final exam, review the content related to Chapters 6 -10 in:

- your textbook
- the [COP 2000 Glossary](#) (on the class web site)
- the [Example pages](#) (on the class web site)

**Be prepared to answer questions about the use of the following operators (as described in Chapters 6&9):**

& - when used as a [unary](#) operator in front of a *variable*, it refers to the [address](#) of a storage [location](#).

& - when used as a unary operator in front of a [formal parameter](#), it declares it as a [reference variable](#).

\* - when used as a unary operator in front of a variable, it [dereferences](#) the identifier so that it refers to the memory location indicated by the contents of the variable rather than directly to the contents of the variable.

**And remember:** Any reference to an array identifier without using a bracketed subscript will be interpreted by C++ as meaning "the *address* of the bottom (subscript 0) element of that array". (See Chap.7)

---

**You will be asked to describe syntax (not run-time) errors in C++ source code statements. The statement might contain no syntax errors; in which case you should answer "OK". Some examples with answers follow:**

SOURCE: `void FuncOne (int A, B); // Function prototype for FuncOne`

ANSWER: The second formal parameter (B) is missing its data type.

SOURCE: `cout << "Address: " << &A << endl; // Display the address of variable A`

ANSWER: OK

SOURCE: `MyFunc ( A ); // Call function MyFunc passing it entire array A`

ANSWER: OK

SOURCE: `float F{5}; // Allocate a floating point array with 5 elements`

ANSWER: The size should be enclosed in brackets, not braces.

SOURCE: `B(5)=2; // Assign 2 into element 5 of array B`

ANSWER: The subscript should be enclosed in brackets, not parentheses.

SOURCE: `vector<int> V(5); // Allocate a floating point vector with 5 elements`

ANSWER: OK here, however subscripts are written using [brackets] rather than (parentheses) when *referencing* elements.

---

**Know how to write and recognize valid code to perform each of the following tasks:**

- declare a function prototype (uses semi-colon) for a function defined following the function that calls it.
- define and then call a function that will not return any data (should have a void data type) and receives no data when called (should have an EMPTY formal parameter list and an EMPTY actual parameter list).
- define and then call a function that returns one value (through the function label) and receives no data when called (should have an EMPTY formal parameter list and an EMPTY actual parameter list).
- define and then call a function that does not return any data, but which expects to receive data when called (should have a non-void formal parameter list with data types and an appropriate actual parameter list).
- define and then call a function that returns one value (through the function label) and which expects to receive data when called (should have a non-void formal parameter list with data types and an appropriate actual parameter list)
- define a storage location to store a pointer for an integer address.
- define a storage location to store a pointer for a floating point address.
- assign the address of an already defined floating point variable into a pointer variable.
- assign a value to a storage location using an already defined and initialized integer pointer.
- display the address stored in a pointer variable.

- define and then call a function that will pass more than one value back to variables declared by the parent function using reference variables as formal parameters (using the & operator). The actual parameter list will contain variables are "passed by reference". The function should be defined as having a void data type.
- define an array and initialize it (load it with a list of values).
- define and then call a function that will pass more than one value back to variables declared by the parent function using indirection with pointers that were passed in – thus the formal parameter list will define pointers (using the \* indirection operator) and the actual parameter list will contain addresses/pointers (using the & "address of" operator). The function should be defined as having a void data type.
- define an array with space for a specified quantity of unknown elements.
- define an empty vector.
- define and initialize a vector a specified quantity of known elements..
- assign a value into a specified element of an array (such as the bottom element of an array named X).
- assign a value into a specified element of a vector (such as the bottom element of a vector named X).
- append a new element onto the end of a vector.
- display the contents of a specified element of an array (such as the bottom element of an array named X).
- display the entire contents of a specified array in a column using an automatic loop with a counter.
- pass an array into a function.
- call a function that was defined as having a void data type (does not return any data to its parent function) and expects to receive data when called.
- determine the number of elements in a vector.
- remove all elements from a vector.

---

**You will be asked to manipulate the contents of a numeric array, sorting or otherwise rearranging the contents of the elements. For example, the following source code would swap all elements in an integer array from the "bottom" (element with the lowest subscript) to the "top" (highest subscripted element).**

```
#include <iostream>
#include <iomanip>
using namespace std;
#define SIZE 5

int main ()
{
    int C; /* Loop index/counter */
    int T; /* Temporary Swap Variable */
    int A[]={1,2,3,4,5};

    /* Display the array from bottom element to top in a line */
    for (C=0; C<SIZE; C++) cout << setw(3) << A[C]; cout << endl;

    /* Swap all elements of the array from bottom to top */
    for (C=0; C<SIZE/2; C++)
        { T = A[C]; A[C]=A[SIZE-C-1]; A[SIZE-C-1]=T; }

    /* Display the array from bottom element to top in a line */
    for (C=0; C<SIZE; C++) cout << setw(3) << A[C]; cout << endl;

    return 0;
}
```

The output produced by the code above would be:

```
1  2  3  4  5
5  4  3  2  1
```

Examine the identifiers in the C++ source code below on the left and fill in the table on the right.

In the column labeled "Declared as", enter one of the following:

Array Variable, Function, Parameter, Symbolic Constant, or Scalar Variable.

In the "Scope" column, enter either "Global" or name the function in which the label is local.

(Complete the table and answer the questions that follow, then see the next page for the answers.)

Source Code	Label	Declared as	Scope
<pre> #include &lt;iostream&gt; #include &lt;iomanip&gt; using namespace std;  #define A 3  int B=2; int C[]={10,20,30};  void D (int E) { while (E&lt;A)   { cout &lt;&lt; setw(3) &lt;&lt; C[E];     E = E+1; }   cout &lt;&lt; endl; }  float F (int G) { float H;   H = G / 3;   return H; }  void I (int J, int *K, char L[]) { int M;   for (M=0; M&lt;J; M++) L[M]='0';   *K = M-1; }  int main () {   char N[A]="ab";   int P;   D (0);   if ( F(B)==B ) cout &lt;&lt; "Yes";     else cout &lt;&lt;"No";   I (A, &amp;P , N);   cout &lt;&lt; endl &lt;&lt; P &lt;&lt; endl;   return 0; }                 </pre>	A		
	B		
	C		
	D		
	E		
	F		
	G		
	H		
	I		
	J		
	K		
	L		
	M		
	N		
	P		

Would this source code compile? \_\_\_\_\_

How many elements does C have? \_\_\_\_\_

List any label(s) being used as a C++ pointer. \_\_\_\_\_

List any label(s) being used as a subscript. \_\_\_\_\_

Would the following statement be valid in main? \_\_\_\_\_

cout << F(B);

What output would be produced by D? \_\_\_\_\_

Would this program display the word "Yes"? \_\_\_\_\_

What value will P have at the end of the program? \_\_\_\_\_

## Answers to the Questions on the preceding page:

Source Code	Label	Declared as	Scope
#include <iostream>	A	Symbolic Constant	Global
#include <iomanip>	B	Scalar Variable	Global
using namespace std;	C	Array Variable	Global
#define A 3	D	Function	Global
int B=2;	E	Parameter	D
int C[]={10,20,30};	F	Function	Global
void D (int E)	G	Parameter	F
{ while (E<A)	H	Scalar Variable	F
{ cout << setw(3) << C[E];	I	Function	Global
E = E+1; }	J	Parameter	I
cout << endl;	K	Parameter	I
}	L	Parameter	I
float F (int G)	M	Scalar Variable	I
{ float H;	N	Array Variable	main
H = G / 3;	P	Scalar Variable	main
return H;			
}			
void I (int J, int *K, char L[])			
{ int M;			
for (M=0; M<J; M++) L[M]='0';			
*K = M-1;			
}			
int main ()			
{			
char N[A]="ab";			
int P;			
D (0);			
if ( F(B)==B ) cout << "Yes";			
else cout <<"No";			
I (A, &P , N);			
cout << endl << P << endl;			
return 0;			
}			

Would this source code compile? YES

How many elements does C have? 3

List any label(s) declared as a C++ pointer. K (in function I)

List any label(s) being used as a subscript. E and M (but not A, which is being used as a size)

Would the following statement be valid in main? YES (because F(B) returns a float)

cout << F(B);

What output would be produced by D?  10 20 30 (where    is a blank space)

Would this program display the word "Yes"? NO (because F does integer division assigning 0 to H)

What value will P have at the end of the program? 2 (as a result of indirect assignment in I)