Passing and receiving parameters

		Function declaration	Caller	Function definition	Who sees changes made to the variable by the function?
Passing a variable	By value	void foo (int);	foo (a);	void foo (int x)	Changes to x are local to function and not seen by caller. e.g. x++; increments x within the function only.
	By reference	void foo (int&);	foo (a);	void foo (int &x)	Changes the value of x are seen by both the function and caller. e.g. x++; increments x for both function and caller.
Passing an array	By reference	void foo (int[]);	foo (anArray);	void foo (int theArray[])	Changes to the value of theArray[i] are seen by both the function and the caller. e.g. theArray[i]++; increments theArray[i] for both function and caller.

Passing a pointer	By value	void foo (int*);	foo (aPtr);	void foo (int* thePtr)	Changes to thePtr are local to the function and not seen by caller.
					e.g. thePtr=&something;
					changes only the function's local copy of thePtr.
					e.g. *thePtr++;
					increments the integer pointed to by thePtr for both function and caller.

	By reference	void foo (int*&);	foo (aPtr);	void foo (int*&thePtr)	Changes to thePtr are seen by both the function and the caller.
					e.g. ptr=&thePtr;
					changes thePtr for both the function and caller
					e.g. *thePtr++;
					increments the integer pointed to by thePtr for both function and caller.

Passing an array of pointers	By reference	void foo (int*[]);	foo (aPtrArray);	void foo (int*ptrArray[])	Changes to ptrArray are seen by both the function and caller. e.g. ptrArray[i] = &something; changes ptrArray[i] for both the function and the caller.
Passing a structure variable	By value	void foo (StructName);	foo (aStructVariable);	void foo (StructName structVariable)	Changes to structVariable are local to function and not seen by caller. e.g. structVariable.memberName = something; changes seen by function only.
	By reference	void foo (StructName&);	foo (aPtr);	void foo (StructName &structVariable)	Changes to the structVariable are seen by the function and caller. e.g. structVariable.memberName = something; changes seen by both function and caller.
Passing a class object	By value	void foo (ClassName);	foo (anObject);	void foo (ClassName object)	Changes made to public or private members are local to the function and not seen by the caller. e.g. objectName.memberName = something; changes seen by function only. Private members are accessible only through getters and setters. Changes to private members seen by function only.
	By reference	void foo (ClassName&);	foo (anObject);	void foo (ClassName &object)	Changes made to public or private members are seen by both the function and caller. e.g. objectName.memberName = something; changes seen by both function and caller. Private members are accessible only through getters and setters. Changes to private members seen by both function and caller.
Passing an array of structure pointers	By reference	Void foo (StructName*[]);	foo (stPtrArray);	void foo (StructName*ptrArray[])	Changes to the structure variables pointed to by the array are seen by the function and caller. Changes to the pointers in the array are seen by the function and caller.

Passing an array of class object pointers	By reference	Void foo (ClassName*[]);	foo (objPtrArray);	void foo (ClassName*ptrArray[])	Changes to the object members pointed to by the array are seen by the function and caller. Changes to the pointers in the array are seen by the function and caller.