Computer Programming & Lab.
Lab. #9 & HW#9
Assignment Date: 2010/11/29
Due Date: 2010/12/6 開始講課前至於講桌上，交答案即可。

7.1 Introduction

7.1. Which statement about pointers is false?
a) They are among C’s most difficult capabilities to master.
b) They enable programs to simulate call by reference.
c) They are useful to create and manipulate dynamic data structures.
d) They cannot be used to create data structures that do not grow and shrink at execution time.

7.2 Pointer Variable Definitions and Initialization

7.2 Pointers can not be used to
(a) find the address of a variable in memory
(b) reference values directly
(c) simulate call-by-reference
(d) manipulate dynamic data structures

7.3 Pointers may be assigned to which of the following?
(a) all integer values
(b) an address
(c) NULL
(d) both (b) and (c)

7.4. Pointers are variables that contain __________ as their values
a) strings
b) flowlines
c) memory addresses
d) directions

7.5. A non-pointer variable name __________ references a value and a pointer variable name __________ references a value.
a) directly, directly
b) directly, indirectly
c) indirectly, directly
d) indirectly, indirectly

7.6. The definition
    int *count;
a) is a syntax error because only pointers can be defined with * notation
b) is a compile-time error.
c) is a logic error.
d) is a correct definition of integer pointer count.

7.7. Referencing a value through a pointer is called __________.
a) interdiction
b) indirection
c) intermediation
d) redirection

7.8. Which statement about pointers is false.
a) They can be defined to point to objects of any data type.
b) The indirection operator * distributes to all comma-separated variable names in a definition.
c) The letters Ptr in a pointer variable name are optional.
d) A pointer may be initialized to 0, NULL or an address.

7.9. Which statement about pointers is false?
a) A pointer with the value NULL points to nothing.
b) NULL is a symbolic constant defined in the <stdio.h> header file.
c) Initializing a pointer to 0 is equivalent to initializing a pointer to NULL, but NULL is preferred.
d) The values 0 and 1 are the only values that can be assigned directly to a pointer variable.

7.3 Pointer Operators

7.10 The & operator can be applied to
(a) constants
(b) variables defined with the storage class register
(c) variables defined with the storage class static
(d) rvalues

7.11 All of the following could cause a fatal execution-time error except
(a) dereferencing a pointer that has not been assigned to point to a specific address
(b) dereferencing a pointer that has not been initialized properly
(c) dereferencing a 0 pointer
(d) dereferencing a nonpointer

7.12 Three of the following expressions have the same value. Which of the following’s value is different from the others?
(a) *&Ptr
(b) &*Ptr
(c) *Ptr
(d) Ptr

7.13. The __________, or address operator, is a unary operator that returns the address of
its operand. 
  a) & 
  b) && 
  c) * 
  d) ** 

7.14. The statement 
    \[ y = &yPtr; \]
  a) assigns the address of the variable \( y \) to pointer variable \( yPtr \).
  b) assigns the address of the variable \( yPtr \) to pointer variable \( y \).
  c) is a compilation error.
  d) is a logic error.

7.15. Which statement is false? 
  a) The operand of the address operator must be a variable.
  b) The address operator cannot be applied to constants or to expressions.
  c) The address operator can be applied to variables defined with the storage class `register`.
  d) The address operator can be applied to variables defined with the storage class `static`.

7.16. Which statement is false? 
  a) The unary * operator is called the indirection operator or the dereferencing operator.
  b) The operand of the unary * operator must be a pointer.
  c) The unary * operator returns the value of its operand.
  d) Placing a * operator before a pointer dereferences the pointer.

7.17. Which statement is generally false? 
  a) Initializing pointers is optional.
  b) Dereferencing an uninitialized pointer could lead to a fatal execution time error.
  c) Dereferencing an uninitialized pointer could accidentally modify important data.
  d) Dereferencing an uninitialized pointer causes a syntax error.

7.18. The unary * and __________ are complements of one another. 
  a) / 
  b) ^ 
  c) & 
  d) | 

### 7.4 Passing Arguments to Functions by Reference

7.19 Which of the following is not a valid way to pass arguments to a function in C? 
  (a) call-by-reference with reference arguments 
  (b) call-by-value 
  (c) call-by-reference with pointer arguments 
  (d) call-by-value with pointer arguments
7.20 When a compiler encounters a function parameter for a single-subscripted array of the form `int a[]`, it converts the parameter to
(a) `int a`
(b) `int &a`
(c) `int * a`
(d) `int * const a`

7.21. Which statement is false.
a) All function calls in C pass arguments call-by-value.
b) Simulated call-by-reference enables a called function to modify variables in the calling function.
c) Call-by-value is always more efficient than simulated call-by-reference.
d) In C, programmers use pointers and the indirection operator to simulate call-by-reference.

7.22. When calling a function with arguments that should be modified, the __________ of those arguments are passed.
a) memory
b) addresses
c) values
d) complements

7.23. If array name `arrayName` is passed to a function, C automatically passes __________.
(a) `&arrayName[ 0 ]`
(b) `arrayName[ 1 ]`
(c) `arrayName[ 0 ]`
(d) `*arrayName`

7.24. When the __________ of a variable is passed to a function, the indirection operator (*) may be used in the function to modify the __________ at that location in the caller’s memory.
a) address, address  
b) value, address  
c) value, value  
d) address, value

7.25. Which statement is false?
a) It is necessary to include names of pointer arguments in function prototypes. 
b) A function receiving an address as an argument must define a pointer parameter to receive the address. 
c) The compiler does not distinguish between a function that receives a pointer and a function that receives a single-subscripted array. 
d) The function must “know” whether it is receiving a single-subscripted array or simply a single variable for which it is to perform simulated call by reference.
Using the const Qualifier with Pointers

7.26 A function that modifies an array by using pointer arithmetic to process every value should have a parameter that is
(a) a nonconstant pointer to nonconstant data
(b) a nonconstant pointer to constant data
(c) a constant pointer to nonconstant data
(d) a constant pointer to constant data

7.27 A function that prints a string should have a parameter that is
(a) a nonconstant pointer to nonconstant data
(b) a nonconstant pointer to constant data
(c) a constant pointer to nonconstant data
(d) a constant pointer to constant data

7.28 An array name is
(a) a nonconstant pointer to nonconstant data
(b) a nonconstant pointer to constant data
(c) a constant pointer to nonconstant data
(d) a constant pointer to constant data

7.29 What method should be used to pass an array to a function that does not modify the array and only looks at it using array subscript notation?
(a) a constant pointer to constant data
(b) a constant pointer to nonconstant data
(c) a nonconstant pointer to constant data
(d) a nonconstant pointer to constant data

7.30. Which statement is false?
a) The const qualifier enables the programmer to inform the compiler that the value of a particular variable should not be modified.
b) The const qualifier did not exist in early versions of C; it was added to the language by the ANSI C committee.
c) Using the const qualifier typically violates the principle of least privilege
d) Adding const to C programs that do not use it presents significant opportunities for improvement in the software engineering of that code.

7.31. Which of the following most closely reflects the principle of least privilege.
a) Always grant a function at least enough access to the data in its parameters to accomplish its specified task.
b) Always grant a function enough access to the data in its parameters to accomplish its specified task, but no more.
c) Always grant a function the least amount of access to the data in its parameters.
d) Never grant a function access to the data in its parameters.

7.32. Which statement is false?
a) If a value should not change in the body of a function to which it is passed, the value should be defined `const` to ensure that it is not accidentally modified.
b) Attempts to modify the value of a variable defined `const` are caught at execution time.
c) One way to pass a pointer to a function is to use a non-constant pointer to non-constant data.
d) It is dangerous to pass a non-pointer into a pointer argument.

7.33. The highest level of data access is granted by a
a) non-constant pointer to non-constant data.
b) non-constant pointer to constant data.
c) constant pointer to non-constant data.
d) constant pointer to constant data.

7.34. Which statement is false?
a) Arrays and structures are aggregate data types.
b) Structures in C are sometimes called records in other languages.
c) A structure is capable of storing many related data items of different data types.
d) Structures are always passed call by reference.

7.35. When a structure must be passed to a function, we can use pointers to constant data to get the performance of a call by _________ and the protection of a call by _________.
   a) value, value
   b) value, reference
   c) reference, value
   d) reference, reference

7.36. The least access privilege is granted by a _________ pointer to _________ data.
a) non-constant, non-constant
b) non-constant, constant
c) constant, non-constant
d) constant, constant

7.6 Bubble Sort Using Call-by-Reference

7.37. Which statement is false?
a) The notations `int *array` and `int array[]` are interchangeable.
b) Function prototypes may not be placed inside functions.
c) Good advice: To encourage software reusability, when passing an array, also pass the size of the array.
d) Global variable violate the principle of least privilege.
7.7 **sizeof Operator**

7.38 `sizeof`
(a) is a binary operator
(b) returns the total number of elements in an array
(c) usually returns a double
(d) returns the total number of bytes in an array

7.39 Which of the following gives the number of elements in the array `int r[]`?
(a) `sizeof ( r )`
(b) `sizeof ( *r )`
(c) `sizeof ( r ) / sizeof ( int )`
(d) `sizeof ( *r ) / sizeof ( int )`

7.40. Which statement is false?

a) Using global variables, instead of passing arguments, can increase program performance, but reduce effective software engineering.
b) Hardcoding into a function the size of an array that will be passed as an argument rather than passing the size of the array as a parameter, reduces the software reusability of that function.
c) `sizeof` is an operator that calculates the size of its argument in bytes at compile time.
d) When applied to the name of an array, `sizeof` returns the number of elements in the array.

7.41. An expression such as
```
    sizeof( arrayName ) / sizeof( double )
```
might typically be used to determine
a) the size of an array
b) the number of elements in an array
c) the number of elements in half an array
d) the size of an element of an array

7.8 **Pointer Expressions and Pointer Arithmetic**

7.42 Which of the following can have a pointer as an operand?
(a) `++`
(b) `*=`
(c) `%`
(d) `/`

7.43 Given that `k` is an integer array starting at location 2000, `kPtr` is a pointer to `k`, and each integer is stored in 4 bytes of memory, what location does `kPtr + 3` point to?
(a) 2003
(b) 2006
(c) 2012
(d) 2024
7.44 A pointer can not be assigned to
(a) another pointer of the same type
(b) a pointer to void
(c) a pointer of a type other than its own type and void
(d) any other pointer by using the cast operator

7.45 Comparing pointers and performing arithmetic on them is meaningless unless
(a) they point to members of the same array
(b) they point to arrays of the same type
(c) they point to arrays of equal size
(d) they point to different locations

7.46. Which statement is false?
(a) A pointer may be incremented or decremented.
(b) An integer may be added to a pointer.
(c) All operators normally used in arithmetic expressions, assignment expressions and comparison expressions are valid in conjunction with pointer variables.
(d) A pointer may not be added to a pointer.

7.47. Which statement is false?
(a) A pointer can always be assigned to another pointer of the same type.
(b) A cast operator must always be used to convert the pointer on the right of an assignment to the pointer type on the left of the assignment.
(c) Variables of all pointer types can be assigned a pointer to void.
(d) A pointer to void can be assigned a pointer of any type.

7.9 Relationship Between Pointers and Arrays

7.48 Assuming that t is an array and tPtr is a pointer to that array, what expression refers to the address of the fourth element?
(a) *( tPtr + 3 )
(b) tPtr[ 3 ]
(c) &t[ 3 ]
(d) *( t + 3 )

7.49 Consider the following function:
void reverse( char * string1, const char * string2 )
{
    int stringsize = sizeof( string1 )/sizeof( char );
    *( string1 + stringsize - 1 ) = '\0';
    string1 = string1 + stringsize - 2;
    for ( ; *string2 != '\0'; string1--, string2++ )
        *string1 = *string2;
}

What method does the function use to refer to array elements?
(a) array/subscript notation
(b) array/offset notation
(c) pointer/subscript notation
(d) pointer/offset notation

7.50. If bPtr is assigned b (the name of an array), then array element b[ 3 ] can alternatively be referenced with the pointer expression __________.
a) bPtr + 3
b) b[ bPtr + 3 ]
c) *b [ bPtr + 3 ]
d) *( bPtr + 3)