CIENCIA DE LA COMPUTACION LABORATORIO C

Leonardo Daniel Valdivia Ramos

8.1 Responde a cada uno de las siguientes:
a) Un puntero es una variable que contiene el valor de una direccion de otra variable.
b) Un puntero se debe inicializar a nullptr o una direccion.
c) El unico entero que puede ser directamente asignado a un puntero es 0 .
8.2 señale si es verdadero o falso.
a) el operador de direcciones solo se puede aplicar a constantes y expresiones.
Falso el operador de direcciones no solo se puede aplicar a constantes
b) Un puntero de tipo void puede ser desreferido.
Falso
c) A pointer of one type can't be assigned to one of another type without a cast
Falso los punteros de cualquier tipo pueden ser asignados a punteros void.
8.3

```
a)numeros[10] = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5,6.6,7.7,8.8,9.9};
b) double *nPtr;
c) for ( int i = 0; i < 10; i++ )
      cout << numeros[ i ] << ' ';
d) nPtr = numeros;
   nPtr = &numeros[0];
e) for ( int i = 0; i < 10; i++ )
      cout << *(numeros+i) << ' ';
cout << *( nPtr + j ) << ' ';
h) numbers[ 3 ]
   *( numbers + 3 )
   nPtr[ 3 ]
   *( nPtr + 3 )
i) la direccion es 1002500 + 8 * 8 = 1002564. el valor es 8.8.
j) la direccion del numero[ 5 ] es 1002500 + 5 * 8 = 1002540.
```

8.4

```
a) double *fPtr = nullptr;
b) fPtr = &number1;
c) cout << "el valor de *fPtr es " << *fPtr << endl;
d) number2 = *fPtr;
e) cout << "The value of number2 es " << number2 << endl;
f) cout << "la direccion de number1 es " << &number1 << endl;
g) cout << "la direccion que contiene fPtr es " << fPtr << endl;
```

8.5

```
a) void intercambio( double *x, double *y )
b) void intercambio( double *, double * );
c) char vocales[] = "AEIOU";
   char vocales[] = { 'A', 'E', 'I', 'O', 'U', '\0' };
```

8.6 Busque el error en el siguiente codigo

```
int *zPtr; // zPtr will reference built-in array z
void *sPtr = nullptr;
int number;
int z[ 5 ] = { 1, 2, 3, 4, 5 };
a) ++zPtr;
Error zPtr no fue inicializado
b) // use pointer to get first value of a built-in array
number = zPtr;
Number no toma el valor sino la direccion
c) // assign built-in array element 2 (the value 3) to number
number = *zPtr[ 2 ];
Error la manera correcta es number = zPtr[ 2 ];
d) // display entire built-in array z
for ( size_t i = 0; i <= 5; ++i )
cout << zPtr[ i ] << endl;
Error hay que cambiar a solo <.
e) // assign the value pointed to by sPtr to number
number = *sPtr;
Hay que desreferenciar el puntero cambiando
number = *static_cast< int * >( sPtr )
f) ++z;
no se puede modificar la direccion del puntero del array
```

8.7 Verdadero o falso

a) dos punteros que apuntan a diferentes valores no se pueden comparar.
Falso

b) porque el nombre de una matriz se puede convertir impplicitamente en un puntero esta puede ser manipulada de la misma manera que los punteros

Falso porque no se puede modificar el puntero del primer elemento
8.8

```
a) Declare a built-in array of type unsigned int called values
   with five elements, and initialize the elements to the even
   integers from 2 to 10. Assume that the constant SIZE has been
    defined as 5.
unsigned int valores[5]={2,4,6,8,10};
b) Declare a pointer vPtr that points to an object of type
   unsigned int.
unsigned int *vPtr=valores;
c) Use a for statement to display the elements of built-in array
    values using array subscript notation.
for ( int i = 0; i < 5; i++ )
     cout << valores[ i ] << ' ';
d) Write two separate statements that assign the starting
   address of built-in array values to pointer variable vPtr.
vPtr=valores;
vPtr=&valores[0];
e) Use a for statement to display the elements of built-in array
    values using pointer/offset notation.
for ( int i = 0; i < 5; i++ )
     cout << vPtr[i] << ' ';
f) Use a for statement to display the elements of built-in array
    values using pointer/offset notation with the built-in
   arrays name as the pointer.
for ( int i = 0; i < 5; i++ )
     cout << *(vPtr+i) << ' ';
g) Use a for statement to display the elements of built-in array
    values by subscripting the pointer to the built-in array.
for ( int i = 0; i < 5; i++ )
     cout << *(valores+i) << ' ';

h) Refer to the fifth element of values using array subscript
   notation, pointer/offset notation with the built-in array
   names as the pointer, pointer subscript notation and
   pointer/offset notation.
cout << vPtr[4];
cout << *(valores+4);
cout << *(vPtr+4);
cout << valores[ 4 ];
```

8.9 For each of the following, write a single statement that performs the specified task. Assume that long variables value1 and value2 have been declared and value1 has been initialized to 200000.

```
a) Declare the variable longPtr to be a pointer to an object of
   type long.
    long longPtr;
b) Assign the address of variable value1 to pointer variable
   longPtr.
    longPtr=&value1;
c) Display the value of the object pointed to by longPtr.
    cout<<*longPtr;
d) Assign the value of the object pointed to by longPtr to
   variable value2.
    value2=*longPtr;
e) Display the value of value2.
    cout << value2;
f) Display the address of value1.
    cout << &value1;
g) Display the address stored in longPtr. Is the address
   displayed the same as value1s?
    cout << longPtr;
    Si
```

8.10 (Function Headers and Prototypes) Perform the task in each of the following statements:

```
a) Write the function header for function zero that takes a long
    integer built-in array parameter bigIntegers and does not
   return a value.
void zero(long int *number)
b) Write the function prototype for the function in part (a).
void zero(long int *)
c) Write the function header for function add1AndSum that takes
   an integer built-in array parameter oneTooSmall and returns
   an integer.
int add1AndSum(int *oneTooSmall)
d) Write the function prototype for the function described in
   part (c).
int add1AndSum(int *)
```

8.11 (Find the Code Errors) Find the error in each of the following segments. If the error can be corrected, explain how.

```
a) int *number;
cout << number << endl;
imprime una direccion nullptr
b) double *realPtr;
long *integerPtr;
integerPtr = realPtr;
no funciona porque apuntan a diferentes tipos de datos
```

```
c) int * x, y;
    x = y;
no funciona porque x solo puede tomar direcciones
d) char s[] = "this is a character array";
    for ( ; *s != '\0'; ++s)
    cout << *s << ' ';
no funciona porque no podemos modificar el puntero de una cadena
     de caracteres
e)  short *numPtr, result;
    void *genericPtr = numPtr;
    result = *genericPtr + 7;
no funciona porque void no apunta a ese tipo de variable
f) double x = 19.34;
double xPtr = &x;
cout << xPtr << endl;
no funciona porque xPtr no esta declarado como un puntero
```

8.13

```
// What does this program do?
#include <iostream>
using namespace std;
void mystery1( char *, const char * ); // prototype
int main()
{
    char string1[ 80 ];
    char string2[ 80 ];
    cout << "Enter two strings: ";
    cin >> string1 >> string2;
    mystery1( string1, string2 );
        cout << string1 << endl;
} // end main
// What does this function do?
void mystery1( char *s1, const char *s2 )
{
    while ( *s1 != '\0' )
    ++s1;
    for ( ; ( *s1 = *s2 ); ++s1, ++s2 )
    ; // empty statement
} // end function mystery1
concatena dos cadenas de caracteres y la asigna a s1
```

8.4

```
// What does this program do?
#include <iostream>
using namespace std;
```

```
int mystery2( const char * ); // prototype
int main()
{
    char string1[ 80 ];
    cout << "Enter a string: ";
    cin >> string1;
    cout << mystery2( string1 ) << endl;
} // end main
// What does this function do?
int mystery2( const char *s )
{
    unsigned int x;
    for ( x = 0; *s != '\0'; ++s )
    ++x;
    return x;
} // end function mystery2
retorna el largo de la cadena de caracteres
```