Universidad Nacional de San Agustín

E.P. De Ciencia de La Computación

C++ Self-Review Exercises Cap. 8

Bedregal Vento, Adrian Rolando

Docente:
Alvaro Henry Mamani Aliaga

12 de octubre de 2018

SELF-REVIEW EXERCISES

1 Answer each of the following:
a) A pointer is a variable that contains as its value the **memory address** of another variable.
b) A pointer should be initialized to `nullptr` or **an address**.
c) The only integer that can be assigned directly to a pointer is **0**.

2 State whether each of the following is *true* or *false*. If the answer is *false*, explain why.
a) The address operator & can be applied only to constants and to expressions. **(False)**. Conversely, this operator must be an *lvalue* and cannot be applied to constants or expresions that have temporary values (*rvalue*).
b) A pointer that is declared to be of type void * can be dereferenced.
c) A pointer of one type can't be assigned to one of another type without a cast operation.

3 For each of the following, write C++ statements that perform the specified task. Assume that double-precision, floating-point numbers are stored in eight bytes and that the starting address of the built-in array is at location 1002500 in memory. Each part of the exercise should use the results of previous parts where appropriate.

a) Declare a built-in array of type double called numbers with 10 elements, and initialize the elements to the values 0.0 , 1.1 , 2.2 , ..., 9.9. Assume that the constant size has been defined as 10.

```
double numbers[10] = {0.0, 1.1, 2.2, 3.3, 4.4,
                      5.5, 6.6, 7.7, 8.8, 9.9};
```

b) Declare a pointer nPtr that points to a variable of type double.

```
double* nPtr;
```

c) Use a for statement to display the elements of built-in array numbers using array subscript notation. Display each number with one digit to the right of the decimal point.

```
for(size_t i = 0; i < 10; ++i)
{
    if(i == 0)
    {
        cout << "0.0" << ' ';
        continue;
    }
    cout << numbers[i] << ' ';
}
```

d) Write two separate statements that each assign the starting address of built-in array numbers to the pointer variable nPtr.

```
nPtr = numbers;
nPtr = &numbers[0];
```

e) Use a for statement to display the elements of built-in array numbers using pointer/offset notation with pointer nPtr.

```
for(size_t i = 0; i < 10; ++i)
    cout << *(nPtr + i) << ' ';
```

f) Use a for statement to display the elements of built-in array numbers using pointer/offset notation with the built-in array's name as the pointer.

```
for(size_t i = 0; i < 10; ++i)
    cout << *(numbers + i) << ' ';
```

g) Use a for statement to display the elements of built-in array numbers using pointer/subscript notation with pointer nPtr.

```
for(size_t i = 0; i < 10; ++i)
    cout << nPtr[i] << ' ';
```

h) Refer to the fourth element of built-in array numbers using array subscript notation, pointer/offset notation with the built-in array's name as the pointer, pointer subscript notation with nPtr and pointer/offset notation with nPtr.

```
numbers[3];
*(numbers + 3);
*(nPtr + 3);
nPtr[3];
```

i) Assuming that nPtr points to the beginning of built-in array numbers , what address is referenced by nPtr + 8 ? What value is stored at that location?

```
// Garbage Data
```

j) Assuming that nPtr points to numbers[5] , what address is referenced by nPtr after nPtr -= 4 is executed? What's the value stored at that location?

```
// 1.1
```

4 For each of the following, write a single statement that performs the specified task. Assume that floating-point variables number1 and number2 have been declared and that number1 has been initialized to 7.3.

a) Declare the variable fPtr to be a pointer to an object of type double and initialize the pointer to nullptr.

```
double* fPtr = nullptr;
```

b) Assign the address of variable number1 to pointer variable fPtr.

```
fPtr = &number1;
```

c) Display the value of the object pointed to by fPtr.

```
cout << *fPtr << endl;
```

d) Assign the value of the object pointed to by fPtr to variable number2.

```
number2 = *fptr;
```

e) Display the value of number2.

```
cout << number2 << endl;
```

f) Display the address of number1.

```
cout << &number1 << endl;
```

g) Display the address stored in fPtr . Is the address displayed the same as that of number1 ?

```
cout << fPtr << endl; // Yes, it is
```

5 Perform the task specified by each of the following statements:

a) Write the function header for a function called exchange that takes two pointers to double-precision, floating-point numbers x and y as parameters and does not return a value.

```
// void exchange(double* x, float* y)
```

b) Write the function prototype for the function in part (a).

```
void exchange(double*, float*);
```

c) Write two statements that each initialize the built-in array of char s named vowel with the string of vowels, “AEIOU”.

```
char vowel[] = "AEIOU";
```

6 Find the error in each of the following program segments. Assume the following declarations and statements:

```
int *zPtr; // zPtr will reference built-in array z
void *sPtr = nullptr;
int number;
int z[ 5 ] = { 1, 2, 3, 4, 5 };
```

```
++zPtr;
// zPtr was not initialized

// use pointer to get first value of a built-in array
number = zPtr;
// The pointer is still a pointer

// assign built-in array element 2 (the value 3) to number
number = *zPtr[ 2 ];
// zPtr is the name of the array, we cannot dereference
// and then use the [] operator

// display entire built-in array z
for ( size_t i = 0; i <= 5; ++i )
    cout << zPtr[ i ] << endl;
// Goes away bounds

// assign the value pointed to by sPtr to number
    number = *sPtr;
// Tries to dereference a void pointer

++z;
// We cannot modify an array name
```