Advanced Programming Concepts with C++

CSI 2372

Tutorial # 2

Solutions for selected exercises from chapter 3

uOttawa

Ahmedou Jreivine

Exercise 3.27

Assuming txt_size is a function that takes no arguments
and returns an int value, which of the following definitions are illegal?
Explain why.
unsigned buf_size = 1024;
(a) int ia[buf_size];
(b) int ia[4 * 7 - 14];
(c) int ia[txt_size()]; (excepted)
(d) char st[11] = "fundamental";

Solution 3.27

```
unsigned buf_size = 1024;

int ia[buf_size];   // illegal, The dimension value is not a constant expression.

int ia[4 * 7 - 14]; // legal

int ia[txt_size()]; // illegal, The dimension value is not a constant expression.

char st[11] = "fundamental";  // illegal, the string's size is 12.
```

Exercise 3.28:

- What are the values in the following arrays?

```
string sa[10];
int ia[10];

int main() {
    string sa2[10];
    int ia2[10];
}
```

Solution 3.28:

```
string sa[10];      //all elements are empty strings
int ia[10];         //all elements are 0

int main()
{
    string sa2[10]; //all elements are empty strings
    int ia2[10];    //all elements are undefined
}
```

Exercise 3.29:

- List some of the drawbacks of using an array instead of a vector.

- **Solution:**
  1. Size is fixed at compiling time.
  2. No API as that of vector.
  3. Bug prone.

Exercise 3.30:

Identify the indexing errors in the following code:

```
constexpr size_t array_size = 10;
int ia[array_size];
for (size_t ix = 1; ix <= array_size; ++ix)
ia[ix] = ix;
```

Solution 3.30:

```
int main() {
constexpr size_t array_size = 10;
int ia[array_size];  // index from `0` to `array_size - 1`
 // for (size_t ix = 1; ix <= array_size; ++ix)
 //  ia[ix] = ix;

for (size_t ix = 0; ix < array_size; ++ix)
        ia[ix] = ix + 1;

return 0;
}
```

Exercise 3.31:

- Write a program to define an array of ten ints. Give each element the same value as its position in the array.

Solution:

```
#include <iostream>
using std::cout; using std::endl;
int main()
{
    int arr[10];
    for (auto i = 0; i < 10; ++i) arr[i] = i;
    for (auto i : arr) cout << i << " ";
    cout << endl;
    return 0;
}
```

Exercise 3.32:

- Copy the array you defined in the previous exercise into another array. Rewrite your program to use vectors.
- **Solution:**

```
int main(){
    int arr[10];                // array
    for (int i = 0; i < 10; ++i)   arr[i] = i;
    int arr2[10];
    for (int i = 0; i < 10; ++i)   arr2[i] = arr[i];
    vector<int> v(10);             // vector
    for (int i = 0; i != 10; ++i)  v[i] = arr[i];
    vector<int> v2(v);
    for (auto i : v2)        cout << i << " ";
    cout << endl;
    return 0;
}
```

Exercise 3.33:

- What would happen if we did not initialize the scores array in the program on page 116?

- **Solution:**
  - If the scores array was defined inside a function, then the value of each element is undefined.
  - If the scores array was defined outside any function, then the value of each element is 0.

Exercise 3.34:

- Given that p1 and p2 point to elements in the same array, what does the following code do? Are there values of p1 or p2 that make this code illegal?

```
p1 += p2 - p1;
```

Solution:

- It moves p1 with the offset p2 - p1.
- After this statement, p1 and p2 points to the same address.
- Any legal value p1, p2 make this code legal.

Exercise 3.35:

- **Exercise 3.35:** Using pointers, write a program to set the elements in an array to zero.
- **Solution:**

```
#include <iostream>
using std::cout; using std::endl;
int main()
{
    const int size = 10;
    int arr[size];
    for (auto ptr = arr; ptr != arr + size; ++ptr) *ptr = 0;
    for (auto i : arr) cout << i << " ";
    cout << endl;
    return 0;
}
```

Exercise 3.36:

- Write a program to compare two arrays for equality.
- Write a similar program to compare two vectors.

Solution 3.36 1/2:

```
#include <iostream>
#include <vector>
#include <iterator>
using std::begin; using std::end; using std::cout; using std::endl; using std::vector;
// pb point to begin of the array, pe point to end of the array.
bool compare(int* const pb1, int* const pe1, int* const pb2, int* const pe2) {
    if ((pe1 - pb1) != (pe2 - pb2)) // have different size.
        return false;
    else{
        for (int* i = pb1, *j = pb2; (i != pe1) && (j != pe2); ++i, ++j)
         if (*i != *j) return false;
    }
    return true;
}
```

Solution 3.36 2/2:

```
int main()
{
    int arr1[3] = { 0, 1, 2 };
    int arr2[3] = { 0, 2, 4 };
    if (compare(begin(arr1), end(arr1), begin(arr2), end(arr2)))
        cout << "The two arrays are equal." << endl;
    else
        cout << "The two arrays are not equal." << endl;
    cout << "==========" << endl;
    vector<int> vec1 = { 0, 1, 2 };
    vector<int> vec2 = { 0, 1, 2 };
    if (vec1 == vec2)
        cout << "The two vectors are equal." << endl;
    else
        cout << "The two vectors are not equal." << endl;
    return 0;
}
```

Exercise 3.37:

- What does the following program do?

```
const char ca[] = {'h', 'e', 'l', 'l', 'o'};
const char *cp = ca;
while (*cp) {
    cout << *cp << endl;
    ++cp;
}
```

Solution:

- This code will print all characters in ca, afterwards as no \0 appended, UB would happen.
- For most cases, the while loop here won't be terminated as expected and many rubbish would be printed out.

Exercise 3.38:

In this section, we noted that it was not only illegal but meaningless to try to add two pointers. Why would adding two pointers be meaningless?

Solution:

It is similar to adding two addresses such as:

1975 St Laurent Blvd , and

800 King Edward st.

But it is more meaningful to add increment an address to get the next address.

800 King Edward st. ++ ➔

801 King Edward st.

uOttawa

Exercise 3.39:

- Write a program to compare two strings. Now write a program to compare the values of two C-style character strings.
- **Solution 1/2**

```
#include <iostream>
#include <string>
#include <cstring>
using std::cout; using std::endl; using std::string;
int main()
{
    string s1("Mooophy"), s2("Pezy");    // use string.
    if (s1 == s2)
        cout << "same string." << endl;
    else if (s1 > s2)
         cout << "Mooophy > Pezy" << endl;
    else
         cout << "Mooophy < Pezy" << endl;
```

Solution 3.39:

- **Solution 2/2:** use C-Style character strings.

```
    const char* cs1 = "Wangyue";
    const char* cs2 = "Pezy";
    auto result = strcmp(cs1, cs2);
    if (result == 0)
        cout << "same string." << endl;
    else if (result < 0)
        cout << "Wangyue < Pezy" << endl;
    else
        cout << "Wangyue > Pezy" << endl;
    return 0;
}
```

Solution 3.40:

- Write a program to define two character arrays initialized from string literals.
- Now define a third character array to hold the concatenation of the two arrays. Use strcpy and strcat to copy the two arrays into the third.

Solution 3.40:

```
#include <iostream>
#include <cstring>
const char cstr1[] = "Hello";
const char cstr2[] = "world!";
int main()
{
    constexpr size_t new_size = strlen(cstr1) + strlen(" ") + strlen(cstr2) + 1;
    char cstr3[new_size];
    strcpy(cstr3, cstr1);
    strcat(cstr3, " ");
    strcat(cstr3, cstr2);
    std::cout << cstr3 << std::endl;
}
```

Exercise 3.43:

- Write three different versions of a program to print the elements of ia.
  - One version should use a range for to manage the iteration, the other two should use an ordinary for loop in one case using subscripts and in the other using pointers.
  - In all three programs write all the types directly. That is, do not use a type alias, auto, or decltype to simplify the code.

Solution 3.43 2/2:

```
#include <iostream>
using std::cout; using std::endl;
int main()
{
    int arr[3][4] ={{ 0, 1, 2, 3 }, { 4, 5, 6, 7 }, { 8, 9, 10, 11 }};
    for (const int(&row)[4] : arr)          // range for
        for (int col : row)          cout << col << " ";
    cout << endl;
    for (size_t i = 0; i != 3; ++i)          // for loop using subscript
        for (size_t j = 0; j != 4; ++j)   cout << arr[i][j] << " ";
    cout << endl;
    for (int(*row)[4] = arr; row != arr + 3; ++row)      // using pointers.
        for (int *col = *row; col != *row + 4; ++col) cout << *col << " ";
    cout << endl;
    return 0;
}
```

Refereces

Accreditation:

- This presentation is prepared/extracted from the following resources:
  - C++ Primer, Fifth Edition.
    Stanley B. Lippman Josée Lajoie Barbara E. Moo
  - https://github.com/jaege/Cpp-Primer-5th-Exercises
  - https://github.com/Mooophy/Cpp-Primer

uOttawa