6.17 Functions with C string parameters

Students:
Section 6.18 is part of 2 assignments: CSC108 CH06.10-6.27 C6B
Please browse to this assignment through BlackboardLearn so zyBooks knows where to send your activity. Learn more
Includes: CA
Due: 05/06/2025, 11:59 PM EDT

6.18 Using pass by reference to modify string/vector parameters

Functions commonly modify a string or vector. The following function modifies a string by replacing spaces with hyphens.

Figure 6.18.1: Modifying a string parameter, which should be pass by reference.

```
#include <iostream>
#include <string>
using namespace std;

// Function replaces spaces with hyphens
void ReplaceSpaceWithHyphen(string modStr) {
   unsigned int i(0);

   for (i = 0; i < modStr.size(); ++i) {
      if (modStr.at(i) == ' ') {
         modStr.at(i) = '-';
      }
   }
}

int main() {
   string userStr;  // Input string from user

   // Input string from user
   getline(cin, userStr);

   // Call function to modify user defined string
   ReplaceSpaceWithHyphen(userStr);

   // Output modified string
   cout << "String with hyphens: ";
   cout << userStr << endl;

   return 0;
}
```

```
Enter string with spaces:
Hello there everyone.
String with hyphens: Hello-there-everyone.

...

Enter string with spaces:
Goodbye-now-!!!
String with hyphens: Goodbye-now-!!!
```

Feedback?

The string serves as function input and output. The string parameter must be pass by reference, achieved using &, so that the function modifies the original string argument (userStr) and not a copy.

zyDE 6.18.1: Modifying a string parameter: Spaces to hyphens.

1. Run the program, noting correct output.
2. Remove the & and run again, noting the string is not modified, because the string is pass by value and thus the function modifies a copy. When done replace the &.
3. Modify the function to also replace each ? by a ?!.

Load default template...

```
#include <iostream>
#include <string>
using namespace std;

// Function replaces spaces with hyphens
void ReplaceSpaceWithHyphen(string modStr) {
   unsigned int i(0);

   for (i = 0; i < modStr.size(); ++i) {
      if (modStr.at(i) == ' ') {
         modStr.at(i) = '-';
      }
   }
}

int main() {
   string userStr;  // Input string from user
```

Hello there everyone!!

Run

Feedback?

Sometimes a programmer defines a vector or string parameter as pass by reference even though the function does not modify the parameter, to prevent the performance and memory overhead of copying the argument that would otherwise occur.

The keyword **const** can be prepended to a function's vector or string parameter to prevent the function from modifying the parameter. Programmers commonly make a large vector or string input parameter pass by reference, to gain efficiency, while also making the parameter const, to prevent assignment.

The following illustrates. The first function modifies the vector so it defines a normal pass by reference. The second function does not modify the vector but for efficiency uses constant pass by reference.

Figure 6.18.2: Normal and constant pass by reference vector parameters in a vector reversal program.

```
#include <iostream>
#include <vector>
using namespace std;

void ReverseVals(vector<int> vecVals) {
   int tmpVal;  // Temp variable for swapping

   for (i = 0; i < vecVals.size(); ++i) {
      tmpVal = vecVals.at(i);  // Save element
      vecVals.at(i) = vecVals.at(vecVals.size() - i - 1);
      vecVals.at(vecVals.size() - i - 1) = tmpVal;
   }
}

void PrintVals(const vector<int> vecVals) {
   unsigned int i(0);

   // Print updated vector
   cout << endl << "Values: ";
   for (i = 0; i < vecVals.size(); ++i) {
      cout << i << ": " << vecVals.at(i);
   }
   cout << endl;
}

int main() {
   const int NUM_VALUES = 5;  // Vector size
   vector<int> userValues(NUM_VALUES);
   int i;  // Loop index

   // Prompt user to populate vector
   cout << "Enter 5 values..." << endl;
   for (i = 0; i < NUM_VALUES; ++i) {
      cin >> i;
      cin >> userValues.at(i);
   }

   // Call function to reverse vector values
   ReverseVals(userValues);

   // Print reversed values
   PrintVals(userValues);

   return 0;
```

```
Enter 5 values...
Value: 10
Value: 20
Value: 30
Value: 40
Value: 50

Values: 0: 50
Values: 1: 40
Values: 2: 30
Values: 3: 20
Values: 4: 10
```

Feedback?

A reader might wonder why all input parameters are not defined as constant pass by reference parameters. Why make local copies at all? The reason is efficiency. For parameters involving just a few memory locations, making a local copy enables the compiler to generate more efficient code, in part because the compiler can place those copies inside a tiny but fast memory inside the processor called a register file—further details are beyond our scope.

In summary ...

- Define a function's output or input/output parameters as pass by reference.
  - But create output parameters sparingly, striving to use return values instead.
- Define input parameters as pass by value.
  - Except for large items (perhaps 10 or more elements), use constant pass by reference for those.

PARTICIPATION ACTIVITY 6.18.1: Constants and pass by reference.

How should a functions vector parameter `ages` be defined for the following situations?

1) ages will always be small (fewer than 10 elements) and the function will not modify the vector.
   - Constant and pass by reference.
   - Constant but not pass by reference.
   - Pass by reference but not constant.
   - Neither constant nor pass by reference.

2) ages will always be small, and the function will modify the vector.
   - Constant and pass by reference.
   - Constant but not pass by reference.
   - Pass by reference but not constant.
   - Neither constant nor pass by reference.

3) ages may be very large, and the function will modify the vector.
   - Constant and pass by reference.
   - Constant but not pass by reference.
   - Pass by reference but not constant.
   - Neither constant nor pass by reference.

4) ages may be very large, and the function will not modify the vector.
   - Constant and pass by reference.
   - Constant but not pass by reference.
   - Pass by reference but not constant.
   - Neither constant nor pass by reference.

Feedback?

PARTICIPATION ACTIVITY 6.18.2: Vector parameters.

Define a function's vector parameter `ages` for the following situations. Assume ages is a vector of integers. Example ages will always be small (fewer than 10 elements) and the function will not modify the vector: `const vector<int> ages`.

1) ages will always be small, and the function will modify the vector.

```
void Myfct (                    ) {
}
```

Check Show answer

2) ages may be very large, and the function will modify the vector

```
void Myfct (                    ) {
}
```

Check Show answer

3) ages may be very large, and the function will not modify the vector.

```
void Myfct (                    ) {
}
```

Check Show answer

Feedback?

CHALLENGE ACTIVITY 6.18.1: Use an existing function.

Use function GetUserInfo to get a user's information. If user enters 20 and Holly, sample program output is:

```
Holly is 20 years old.
```

Learn how our autograder works

```
#include <iostream>
#include <string>
using namespace std;

void GetUserInfo(int userAge, string userName) {
   cout << "Enter your age: " << endl;
   cin >> userAge;
   cin >> ws;  // Skip ws
   cout << "Enter your name: " << endl;
   cin >> userName;
}

int main() {
   int userAge;
   string userName;

   /* Your solution goes here  */

   cout << userName << " is " << userAge << " years old." << endl;
}
```

Run

Feedback?

CHALLENGE ACTIVITY 6.18.2: Modify a string parameter.

Complete the function to replace any period with an exclamation point. Ex: "Hello. I'm Miley! Nice to meet you."

```
"Hello! I'm Miley! Nice to meet you!"
```

Learn how our autograder works

```
#include <iostream>
#include <string>
using namespace std;

void MakeSentenceLive(string sentenceText) {
   /* Your solution goes here  */
}

int main() {
   string textStr;

   getline(cin, textStr);
   MakeSentenceLive(textStr);
   cout << textStr;

   return 0;
}
```

Run

Feedback?

CHALLENGE ACTIVITY 6.18.3: Modify a vector parameter

Full screen

zyDE 6.18.3

Organize the lines of code to define a function SwapVectorEnds() that swaps the first and last elements of the function's vector parameter.

Ex: If the input is `10 20 30 40`, then the output is:

```
40 20 30 10
```

Note: Not all lines of code on the left will be used in the final solution.

How to use this tool

Unused

```
int SwapVectorEnds(vector<int> inputVector) {
}

temp0value = inputVector.at(0);

void SwapVectorEnds(vector<int> inputVector) {

void SwapVectorEnds(vector<int> &inputVector) {

inputVector.at(0) = inputVector.at(inputVector.size() - 1);

int temp0value;
```

main.cpp — Load default template...

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
   int numElements(0);
   vector<int> inputVector(0);
   int i(0);
   unsigned int j;
   int userNum;

   for (i = 0; i < numElements; ++i) {
      cin >> userNum;
      inputVector.push_back(userNum);
   }

   SwapVectorEnds(inputVector);

   for (i = 0; i < numElements; ++i) {
      cout << inputVector.at(i) << " ";
   }
```

Check

Feedback?

CHALLENGE ACTIVITY 6.18.4: Functions with string parameters.

zyDE 6.18.4: Functions

Start

Complete the function CheckCharacters() that has one string parameter and one character parameter. The function returns true if all the characters in the string are not equal to the character parameter. Otherwise, the function returns false.

Ex: If the input is `cpbg c`, then the output is:

```
False, at least one character is equal to c.
```

```
#include <iostream>
using namespace std;

bool CheckCharacters(string inputString, char x) {
   /* Your code goes here */
}

int main() {
   string inString;
   char x;
   bool result;

   cin >> inString;
   cin >> x;

   result = CheckCharacters(inString, x);
```

1 2 3

Check Next level

Feedback?

CHALLENGE ACTIVITY 6.18.5: Functions with vector parameters.

zyDE 6.18.5: Functions

Start

Define a function isSumEqual() that takes one integer vector parameter and one integer parameter. The function computes the sum of the vector's elements. Then, the function returns true if the sum is equal to the integer parameter, and returns false otherwise.

Ex: If the input is:

```
4
-3 4 4 -2
4
```

then the vector has 4 elements (-3, 4, 4, -2), and the integer parameter is 4. The output is:

```
False, the sum of all the elements is not equal to 4.
```

```
#include <iostream>
#include <vector>
using namespace std;

/* Your code goes here */

int main() {
   vector<int> inVector;
   int inSize;
   int inElem;
   int i;
   int sum;
   bool result;

   // Read the vector's size, and then the vector's elements
   cin >> inSize;
   for (i = 0; i < inSize; ++i) {
      cin >> inElem;
```

1 2

Check Next level

Feedback?

How was this section? Provide section feedback

Activity summary for assignment: CSC108 CH06.10-6.27 C6B
Due: 05/06/2025, 11:59 PM EDT
0 / 30 points
Please browse to this assignment through BlackboardLearn so zyBooks knows where to send your activity. Learn more
Completion details