Laboratory 10A

CS-102

Spring 2022

Laboratory 10: Using C-String and String Classes

- Write a program that has an array of 11 string objects that hold people's names and phone numbers. Use the following data:
  - "Alejandra Cruz, 555-1223",
  - "Joe Looney, 555-0097",
  - "Geri Palmer, 555-8787",
  - "Li Chen, 555-1212",
  - "Holly Gaddis, 555-8878",
  - "Sam Wiggins, 555-0998",
  - "Bob Kain, 555-8712",
  - "Tim Haynes, 555-7676",
  - "Warren Gaddis, 555-9037",
  - "Jean James, 555-4939",
  - "Ron Palmer, 555-2783"

Laboratory 10 - Part 1: Using C-String and String Classes

- The program should ask the user to enter a name or partial name to search for in the array.
  - Call your program: *YourName*-Lab10A-1.cpp
- Any entries in the array that match the string entered should be displayed.
- For example, if the user enters “Palmer” the program should display the following names from the list:
  - Geri Palmer, 555-8787
  - Ron Palmer, 555-2783
- This can be done using the C-String Class.
  - Program 10-6, shown in the appendix to this Lab, shows a similar example, indicating how this might be done using the C-String Class.
  - Using Program 10-6, modify it so that it is able to do a search for the names in our contact list.
  - Note that the Array used here is a two dimensional Array of char
    - The first dimension (row dimension) represents the number of contacts in the Array
    - The second dimension contains the Array of char holding the Contact information for one person
- Once you have that working:
  - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 10 - Part 2: Using C-String and String Classes

- If you wrote Part 1 using the **break** command, rewrite it so that you avoid using the **break** command.
- Call your new program: *YourName*-Lab10A-2.cpp
- Demonstrate your new implementation
  - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 10 – Part 3: Using C++ String and String Classes

- This can also be done using the C++ String Class.
  - For that you will want to make use of the find() method shown on page 599 of Starting Out With C++, 9th Edition, and also shown in the appendix to this lab.
  - Tip: If a string Member Function fails to perform the procedure it was asked to do, it returns -1 to the program calling the function.
- Repeat this search using the C++ **String** Class instead. You may find that you will be able to execute this program using fewer steps than those used for the C-String class.
  - Note that the Array used here is a one dimensional Array of Strings.
  - Note also that you will be including the **string** Class instead of the **cstring** Class.
- Call your program: *YourName*-Lab10A-3.cpp
- Again, when you have this version working OK:
  - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 10 - Part 4: Using C++ String and String Classes

- If you wrote Part 3 using the **break** command, rewrite it so that you avoid using the **break** command.
- Call your program: *YourName*-Lab10A-4.cpp
- Demonstrate your new implementation by:
  - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

Laboratory 10 - Part 5: Using C++ String and String Classes

- Program 10-24 (see appendix D) gives you an algorithm for inserting commas into numbers which are greater than 999.
- This algorithm was designed to deal with currency and assumes that there will be a decimal place in the number, which you type in, and it will put a $ in front of the result.
- You are to modify this program so that its function accepts an integer input, which it then converts to a string (in which form it needs to be if the number is to contain commas in its output).
  - Note: There should be no dollar sign in the output display.
  - You will want to make use of the function: **to_string()**, found in Table 10-6, which converts an integer to a string.
    - to_string(int value); Accepts an int argument and returns that argument converted to a string object.
- Here are some numbers to type in and convert to strings with commas added (if appropriate).
  - 1000
  - 999
  - 2147483647
  - -2147483648
- Call your program: *YourName*-Lab10A-5.cpp
- Demonstrate your resulting function by:
  - If you are doing this Lab synchronously, call the instructor so that you may receive credit for having accomplished this.
  - If you are doing this Lab asynchronously, submit your program to Canvas.

```
// This program uses the strstr function to search an array of char.
#include <iostream>
#include <cstring>      // For strstr
using namespace std;
int main()
{
   // Constants for array lengths
   const int NUM_PRODS = 5;   // Number of products
   const int LENGTH = 27;     // String length
   // Array of products
   char products[NUM_PRODS][LENGTH] =
                { "TV327 31 inch Television",
                  "CD257 CD Player",
                  "TA677 Answering Machine",
                  "CS109 Car Stereo",
                  "PC955 Personal Computer" };

   char lookUp[LENGTH];              // To hold user's input
   char *strPtr = nullptr;           // To point to the found product
   int index;                        // Loop counter
   bool found;                       // Found flag
```

Appendix A:

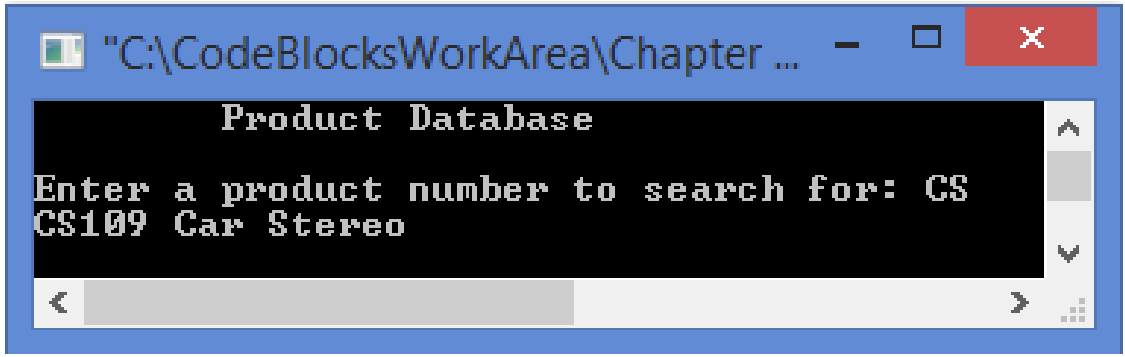
Program10-6: Use of strstr Function

Program10-6 Concluded

```
// Prompt the usr for a product number.
  cout << "\tProduct Database\n\n";
  cout << "Enter a product number to search for: ";
  cin.getline(lookUp, LENGTH);
// Search the array for a matching substring
  for (index = 0; index < NUM_PRODS; index++)
  {
            strPtr = strstr(products[index], lookUp);
            if (strPtr != nullptr)
                        break;
  }

  // If a matching substring was found, display the product info.
  if (strPtr != nullptr)
            cout << products[index] << endl;
  else
            cout << "No matching product was found.\n";

  return 0;
}
```



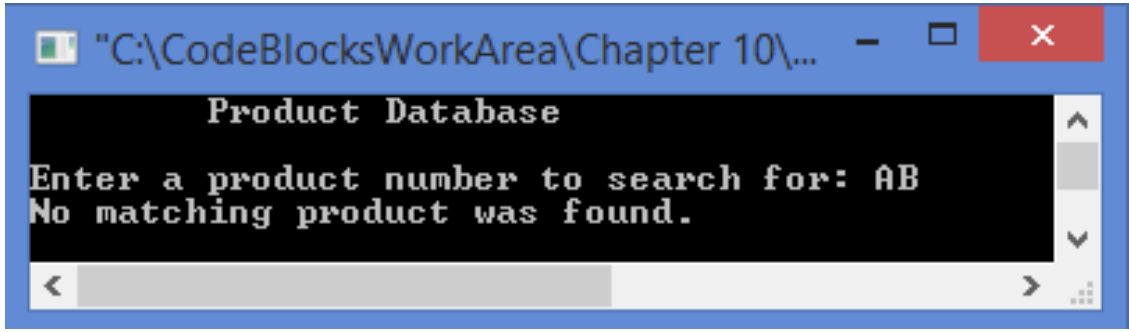


Table 10-8

Member Function Example	Description
`mystring.begin();`	Returns an iterator pointing to the first character in the `string`. (For more information on iterators, see Chapter 16.)
`mystring.c_str();`	Converts the contents of `mystring` to a C-string, and returns a pointer to the C-string.
`mystring.capacity();`	Returns the size of the storage allocated for the `string`.
`mystring.clear();`	Clears the `string` by deleting all the characters stored in it.
`mystring.compare(str);`	Performs a comparison like the `strcmp` function (see Chapter 4), with the same return values. `str` can be a `string` object or a character array.
`mystring.compare(x, n, str);`	Compares `mystring` and `str`, starting at position `x`, and continuing for `n` characters. The return value is like `strcmp`. `str` can be a `string` object or character array.
`mystring.copy(str, x, n);`	Copies the character array `str` to `mystring`, beginning at position `x`, for `n` characters. If `mystring` is too small, the function will copy as many characters as possible.
`mystring.empty();`	Returns true if `mystring` is empty.
`mystring.end();`	Returns an iterator pointing to the last character of the string in `mystring`. (For more information on iterators, see Chapter 16.)
`mystring.erase(x, n);`	Erases `n` characters from `mystring`, beginning at position `x`.
`mystring.find(str, x);`	Returns the first position at or beyond position `x` where the string `str` is found in `mystring`. `str` may be either a `string` object or a character array.
`mystring.find('z', x);`	Returns the first position at or beyond position `x` where '`z`' is found in `mystring`.
`mystring.front();`	Returns the first character in the `string`. (This member function was introduced in C++ 11.)
`mystring.insert(x, n, 'z');`	Inserts '`z`' `n` times into `mystring` at position `x`.
`mystring.insert(x, str);`	Inserts a copy of `str` into `mystring`, beginning at position `x`. `str` may be either a `string` object or a character array.
`mystring.length();`	Returns the length of the string in `mystring`.
`mystring.replace(x, n, str);`	Replaces the `n` characters in `mystring` beginning at position `x` with the characters in string object `str`.
`mystring.resize(n, 'z');`	Changes the size of the allocation in `mystring` to `n`. If `n` is less than the current size of the string, the string is truncated to `n` characters. If `n` is greater, the string is expanded and '`z`' is appended at the end enough times to fill the new spaces.
`mystring.size();`	Returns the length of the string in `mystring`.
`mystring.substr(x, n);`	Returns a copy of a substring. The substring is `n` characters long and begins at position `x` of `mystring`.
`mystring.swap(str);`	Swaps the contents of `mystring` with `str`.

Appendix B: Methods belonging to the String Class Of particular interest:

mystring.find(str, x);

- Returns the first position at or beyond position x where the string **str** is found in mystring.
- **str** may be either a string object or a character array.

```
// This program uses the find function to search an array of String.
#include <iostream>
#include <string>
using namespace std;

int main()
{
   // Constants for array lengths
   const int NUM_PRODS = 11;   // Number of products
   const int LENGTH = 27;     // String length

   // Array of products
   string products[NUM_PRODS] =
                        {"TV327 31 inch Television",
                        "CD257 CD Player",
                        "TA677 Answering Machine",
                        "CS109 Car Stereo",
                        "PC955 Personal Computer"};
```

Appendix C:

Program 10-6 C++ String Version

```
    string lookUp;                          // To hold user's input
    int index;                              // Loop counter
    bool found = false;                     // Found flag
    const int NOTFOUND = -1;
    cout << "\tContact Database\n\n";
    cout << "Enter a Product Number to search for: ";
    getline(cin,lookUp);
    // Search the array for a matching substring
    for (index = 0; index < SIZE; index++)
    {
        if (stringArray[index].find(str,0)!= NOTFOUND)
        {
                found = true;
                break;
        }
    }
    if (found)
         cout << stringArray[index] << endl;
    else
         cout << str << " not found in table." << endl;
   cin.get();
   return 0;
}
```

Program 10-6 C++ String Version Concl.

```
// This program lets the user enter a number. The
// dollarFormat function formats the number as
// a dollar amount.
#include <iostream>
#include <string>
using namespace std;

// Function prototype
void dollarFormat(string &);

int main ()
{
  string input;

  // Get the dollar amount from the user.
  cout << "Enter a dollar amount in the form nnnnn.nn : ";
  cin >> input;
  dollarFormat(input);
  cout << "Here is the amount formatted:\n";
  cout << input << endl;
  return 0;
}
```

Appendix D:

Program 10-24

This program inserts commas into large numbers, to make them more readable. It’s input must be in a string form.

Part 1 of 2

```
//*********************************************************
// Definition of the dollarFormat function. This function    *
// accepts a string reference object, which is assumed to    *
// to hold a number with a decimal point. The function       *
// formats the number as a dollar amount with commas and     *
// a $ symbol.                                               *
//*********************************************************

void dollarFormat(string &currency)
{
  int dp;

  dp = currency.find('.');   // Find decimal point
  if (dp > 3)                // Insert commas
  {
    for (int x = dp - 3; x > 0; x -= 3)
      currency.insert(x, ",");
  }
  currency.insert(0, "$");   // Insert dollar sign
}
```

Appendix D:

Program 10-24 This program inserts commas into large numbers, to make them more readable. It's input must be in a string form. Part 2 of 2