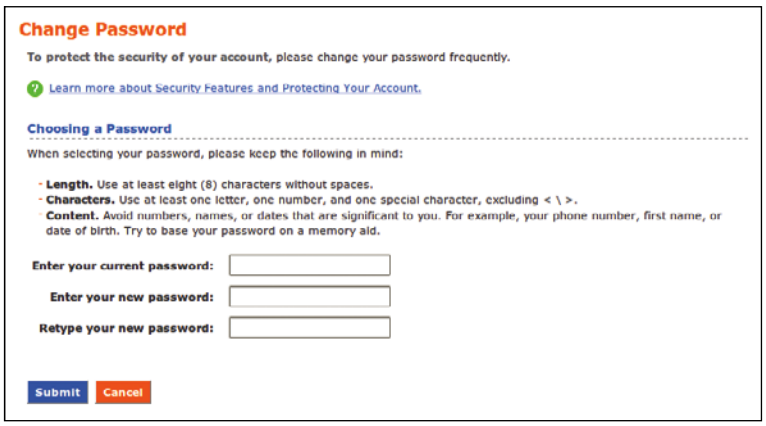
WORKED EXAMPLE 5.1

Generating Random Passwords

Problem Statement Many web sites and software packages require you to create passwords that contain at least one digit and one special character. Your task is to write a program that generates such a password of a given length. The characters should be chosen randomly.



Step 1 Describe what the function should do.

The problem description asks you to write a program, not a function. We will write a password-generating function and call it from the program's `main` function.

Let us be more precise about the function. It will generate a password with a given number of characters. We could include multiple digits and special characters, but for simplicity, we decide to include just one of each. We need to decide which special characters are valid. For our solution, we will use the following set:

```
+ - * / ? ! @ # $ % &
```

The remaining characters of the password are letters. For simplicity, we will use only lowercase letters in the English alphabet.

Step 2 Determine the function's "inputs".

There is just one parameter: the length of the password.

At this point, we have enough information to document the function:

```
/**
   Generates a random password.
   @param length the length of the password
   @return a password of the given length with one digit and one
   special symbol
*/
```

Step 3 Determine the types of the parameter variables and the return value.

The parameter is an integer. The function returns the password, that is, a string. The function will be declared as

```
string make_password(int length)
```

Step 4 Write pseudocode for obtaining the desired result.

Here is one approach for making a password:

Make an empty string called password.
Randomly generate length - 2 letters and append them to password.
Randomly generate a digit and insert it at a random location in password.
Randomly generate a symbol and insert it at a random location in password.

How do we generate a random letter, digit, or symbol? How do we insert a digit or symbol in a random location? In the spirit of stepwise refinement, we will delegate those tasks to helper functions. Each of those functions starts a new sequence of steps, which, for greater clarity, we will place after the steps for this function.

Step 5 Implement the function body.

We need to know the "black box" descriptions of the two helper functions described in Step 4 (which we will complete after this function). Here they are:

```
/**
   Returns a string containing one character randomly chosen from a given string.
   @param characters the string from which to randomly choose a character
   @return a substring of length 1, taken at a random index
*/
string random_character(string characters)

/**
   Inserts one string into another at a random position.
   @param str the string into which another string is inserted
   @param to_insert the string to be inserted
   @return the result of inserting to_insert into str
*/
string insert_at_random(string str, string to_insert)
```

Now we can translate the pseudocode of Step 4 into C++:

```
string make_password(int length)
{
   string password = "";
   for (int i = 0; i < length - 2; i++)
   {
      password = password + random_character("abcdefghijklmnopqrstuvwxyz");
   }
   string random_digit = random_character("0123456789");
   password = insert_at_random(password, random_digit);
   string random_symbol = random_character("+-*/?!@#$%&");
   password = insert_at_random(password, random_symbol);
   return password;
}
```

Step 6 Test your function.

Because our function depends on several helper functions, we must implement the helper functions first, as described in the following sections. (If you are impatient, you can use the technique of stubs that is described in Programming Tip 5.5.)

Here is a simple program that calls the `make_password` function:

```
int main()
{
   srand(time(0));
   string result = make_password(8);
   cout << result << endl;
   return 0;
```

```
}
```

Run the program a few times. Typical outputs are

```
u@taqr8f
i?fs1dgh
ot$3rvdv
```

Each output has length 8 and contains a digit and special symbol. Note that without seeding the random number generator, the program would always print the same result.

Repeat for the First Helper Function

Now it is time to turn to the helper function for generating a random letter, digit, or special symbol.

Step 1 Describe what the function should do.

How do we deal with the choice between letter, digit, or special symbol? Of course, we could write three separate functions, but it is better if we can solve all three tasks with a single function. We could require a parameter, such as 1 for letter, 2 for digit, and 3 for special symbol. But stepping back a bit, we can supply a more general function that simply selects a random character from *any* set. Passing the string `"abcdefghijklmnopqrstuvwxyz"` generates a random lowercase letter. To get a random digit, pass the string `"0123456789"` instead.

Now we know what our function should do. Given any string, it should return a random character in it.

Step 2 Determine the function's "inputs".

The input is any string.

Step 3 Determine the types of the parameter variables and the return value.

The input type is clearly `string`. We want to make a string from the random characters. It is easy to concatenate strings, so we choose to return strings of length 1. The return type is `string`.

The function will be declared as

```
string random_character(string characters)
```

Step 4 Write pseudocode for obtaining the desired result.

random_character(characters)
n = length of characters
r = a random integer between 0 and n - 1
Return the substring of characters of length 1 that starts at r.

Step 5 Implement the function body.

Simply translate the pseudocode into C++:

```
/**
   Returns a string containing one character randomly chosen from a given string.
   @param characters the string from which to randomly choose a character
   @return a substring of length 1, taken at a random index
*/
string random_character(string characters)
{
   int n = characters.length();
   int r = rand() % n;
   return characters.substr(r, 1);
}
```

Step 6 Test your function.

Supply a program for testing this function only:

```
int main()
{
   srand(time(0));
   for (int i = 1; i <= 10; i++)
   {
      cout << random_character("abcdef");
   }
   cout << endl;
   return 0;
}
```

When you run this program, you might get an output such as

```
afcdfeefac
```

This confirms that the function works correctly.

Repeat for the Second Helper Function

Finally, we implement the second helper function, which inserts a string containing a single character at a random location in a string.

Step 1 Describe what the function should do.

Suppose we have a string `"arxcsw"` and a string `"8"`. Then the second string should be inserted at a random location, returning a string such as `"ar8xcsw"` or `"arxcsw8"`. Actually, it doesn't matter that the second string has length 1, so we will simply specify that our function should insert an arbitrary string into a given string.

Step 2 Determine the function's "inputs".

The first input is the string into which another string should be inserted. The second input is the string to be inserted.

Step 3 Determine the types of the parameter variables and the return value.

The inputs are both of type `string`. The return value is also a `string`. We can now fully describe our function:

```
/**
   Inserts one string into another at a random position.
   @param str the string into which another string is inserted
   @param to_insert the string to be inserted
   @return the result of inserting to_insert into str
*/
string insert_at_random(string str, string to_insert)
```

Step 4 Write pseudocode for obtaining the desired result.

There is no predefined function for inserting a string into another. Instead, we need to find the insertion position and then "break up" the first string by taking two substrings: the characters up to the insertion position, and the characters following it.

How many choices are there for the insertion position? If `str` has length 6, there are seven choices:

```
1. |arxcsw
2. a|rxcsw
3. ar|xcsw
4. arx|csw
5. arxc|sw
```

```
6. arxcs|w
7. arxcsw|
```

In general, if the string has length n, there are $n + 1$ choices, ranging from 0 (before the start of the string) to n (after the end of the string). Here is the pseudocode:

```
insert_at_random(str, to_insert)
n = length of str
r = a random integer between 0 and n
Return the substring of str from 0 to r - 1 + to_insert + the remainder of str.
```

Step 5 Implement the function body.

Translate the pseudocode into C++:

```
/**
   Inserts one string into another at a random position.
   @param str the string into which another string is inserted
   @param to_insert the string to be inserted
   @return the result of inserting to_insert into str
*/
string insert_at_random(string str, string to_insert)
{
   int n = str.length();
   int r = rand() % (n + 1);
   return str.substr(0, r) + to_insert + str.substr(r);
}
```

Step 6 Test your function.

Supply a program for testing this function only:

```
int main()
{
   srand(time(0));
   for (int i = 1; i <= 10; i++)
   {
      cout << insert_at_random("arxcsw", "8");
   }
   cout << endl;
   return 0;
}
```

When you run this program, you might get an output such as

```
arxcsw8
ar8xcsw
arxc8sw
a8rxcsw
arxcsw8
ar8xcsw
arxcsw8
a8rxcsw
8arxcsw
8arxcsw
```

The output shows that the second string is being inserted at an arbitrary position, including the beginning and end of the first string.

Here is the complete program:

worked_example_1/password.cpp

```
1 #include <iostream>
2 #include <cstdlib>
3 #include <ctime>
4
```

```
using namespace std;

/**
   Returns a string containing one character randomly chosen from a given string.
   @param characters the string from which to randomly choose a character
   @return a substring of length 1, taken at a random index
*/
string random_character(string characters)
{
   int n = characters.length();
   int r = rand() % n;
   return characters.substr(r, 1);
}

/**
   Inserts one string into another at a random position.
   @param str the string into which another string is inserted
   @param to_insert the string to be inserted
   @return the result of inserting to_insert into str
*/
string insert_at_random(string str, string to_insert)
{
   int n = str.length();
   int r = rand() % (n + 1);
   return str.substr(0, r) + to_insert + str.substr(r);
}

/**
   Generates a random password.
   @param length the length of the password
   @return a password of the given length with one digit and one
   special symbol
*/
string make_password(int length)
{
   string password = "";

   // Pick random letters

   for (int i = 0; i < length - 2; i++)
   {
      password = password + random_character("abcdefghijklmnopqrstuvwxyz");
   }

   // Insert two random digits

   string random_digit = random_character("0123456789");
   password = insert_at_random(password, random_digit);
   string random_symbol = random_character("+-*/?!@#$%&");
   password = insert_at_random(password, random_symbol);
   return password;
}

int main()
{
   srand(time(0));
   string result = make_password(8);
   cout << result << endl;
   return 0;
}
```