WORKED EXAMPLE 7.1

Producing a Mass Mailing

Problem Statement We want to automate the process of producing mass mailings. A typical letter might look as follows:

```
To: Ms. Sally Smith
123 Main Street
Anytown, NY 12345

Dear Ms. Smith:

The Smith family may be the lucky winner in the C++ sweepstakes.
Wouldn't it be exciting if you were the first Anytown residents
to use ACME's new C++ development environment? etc. etc.
```

Another letter with the same template but different values for the variable parts would look like this:

```
To: Mr. Harry Morgan
456 Park Ave
Everyville, KS 67890

Dear Mr. Morgan:

The Morgan family may be the lucky winner in the C++ sweepstakes.
Wouldn't it be exciting if you were the first Everyville residents
to use ACME's new C++ development environment? etc. etc.
```

To set up such a mailing, we start with a template string in which each of the variable parts is denoted by a digit 0 ... 9. (We assume that digits are not otherwise used in the template text.)

```
To: 0 1 2
3
4, 5 6

Dear 0 2:

The 2 family may be the lucky winner in the C++ sweepstakes.
Wouldn't it be exciting if you were the first 4 residents
to use ACME's new C++ development environment? etc. etc.
```

This template is set up in a string, like this:

```
char letter_template[] = "To: 0 1 2\n3\n4, 5 6\n\nDear 0 2: \n\n"
   "The 2 family may be the lucky winner in the C++ sweepstakes.\n"
   "Wouldn't it be exciting if you were the first 4 residents\n"
   "to use ACME's new C++ development environment? etc. etc. \n\n\n";
```

(To declare a literal string in C++ that does not fit on a single line, you write a sequence of literal strings. The compiler combines them into a single string.)

Of course, we could produce a separate string for each mailing, by replacing the digits with the values for a particular recipient. But there is a more efficient way. Your task will be to set up a vector of character pointers that point to successive fragments of the letter, either from the template string or the variable parts. For example, the body of the letter above is represented by pointers to the following strings:

- `": \n\nThe "`
- Variable 2
- `" family may be . . . were the first "`

- Variable 4
- `" residents . . . "`

The variable parts are stored in a two-dimensional array of characters:

```
char variable_parts[10][VAR_LENGTH];
```

Each row of this array contains a variable:

```
strcpy(variable_parts[0], "Ms.");
strcpy(variable_parts[1], "Sally");
strcpy(variable_parts[2], "Smith");
strcpy(variable_parts[3], "123 Main Street");
strcpy(variable_parts[4], "Anytown");
strcpy(variable_parts[5], "NY");
strcpy(variable_parts[6], "12345");
```

Printing a letter is achieved with the following loop:

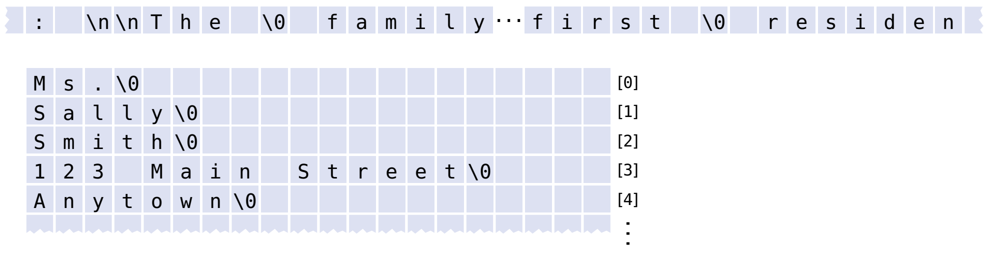
```
vector<char*> fragments = prepare_mailing(letter_template, variable_parts);

for (int i = 0; i < fragments.size(); i++)
{
   cout << fragments[i];
}
```

Your task will be to implement the `prepare_mailing` function, and to produce a `main` function that prints the two letters given above.

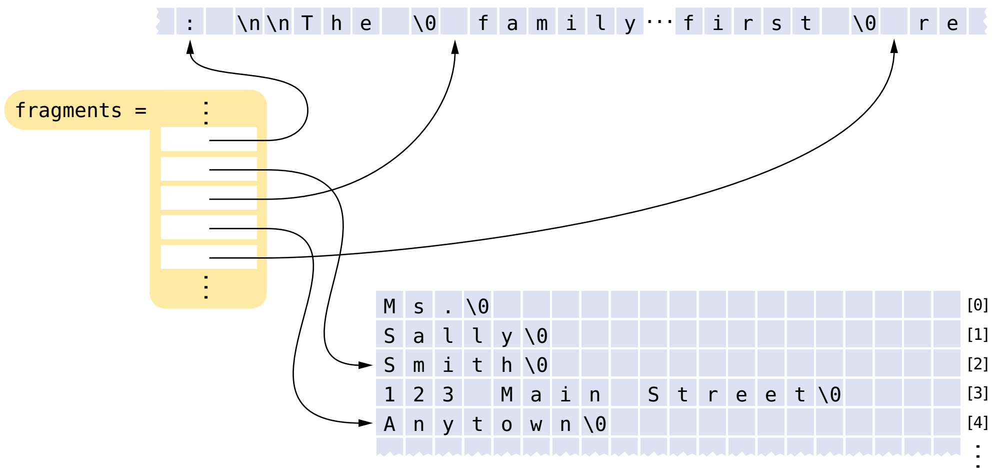
Step 1 Draw a picture.

We start with the data that is being accessed through pointers: the strings in the letter template (after replacing the variable placeholders with null terminators), and the strings in the variable array.



: \n \n T h e \0 f a m i l y ··· f i r s t \0 r e s i d e n
M s . \0 [0]
S a l l y \0 [1]
S m i t h \0 [2]
1 2 3 M a i n S t r e e t \0 [3]
A n y t o w n \0 [4]

The pointers point into the template and variable strings, like this:



: \n \n T h e \0 f a m i l y ··· f i r s t \0 r e
fragments =
M s . \0 [0]
S a l l y \0 [1]
S m i t h \0 [2]
1 2 3 M a i n S t r e e t \0 [3]
A n y t o w n \0 [4]

Step 2 Declare pointer variables.

We collect a sequence of `char*` pointers of unknown length. Therefore, we use a vector of pointers:

```
vector<char*> fragments;
```

Step 3 Initialize the pointers with variable addresses or free store memory.

This is the task of the `prepare_mailing` function. That function needs to scan the letter template for variable indicators, replace them with null terminators, and store pointers in the `fragments` vector. Here is the pseudocode:

```
Set the fragment start to the beginning of the letter template.
For each character in the letter template
    If the character is a digit
        Add the fragment start to the fragments vector.
        Add the row of the variable denoted by the digit to the fragments vector.
        Replace the digit with a null terminator.
        Set the fragment start to the following character.
```

You will find the C++ code in the code listing below.

Step 4 Use the `*` or `[]` operator to access your data.

This step has already been completed in the problem statement. It was a requirement that the fragments can be printed with the following loop:

```
for (int i = 0; i < fragments.size(); i++)
{
   cout << fragments[i];
}
```

All that remains is to complete a `main` program that prints two letters, as shown in the code listing. Note that the `prepare_mailing` function is only called once. Afterward, the variable array is updated for each mailing, but the `fragments` vector never changes.

Here is the complete program:

worked_example_1/mail.cpp

```
#include <cctype>
#include <iostream>
#include <string>
#include <vector>

using namespace std;

const int VAR_LENGTH = 20;

/**
   Prepares a letter template for a mass mailing.
   @param letter a string with placeholders 0 ... 9 (which will be replaced
   with null terminators)
   @param vars a two-dimensional array for holding the variable parts of the letter
   @return a vector of fragments, pointing alternately to strings in the letter
   template and in the variable array
*/
vector<char*> prepare_mailing(char* letter, char vars[][VAR_LENGTH])
{
   vector<char*> fragments;
   char* fragment_start = letter;
   for (char* p = letter; *p != '\0'; p++)
   {
      if (isdigit(*p))
      {
         fragments.push_back(fragment_start);
         fragment_start = p + 1;
         int var_index = *p - '0';
         fragments.push_back(vars[var_index]);
         *p = '\0';
      }
   }
   fragments.push_back(fragment_start);
   return fragments;
}


int main()
{
   char variable_parts[10][VAR_LENGTH];

   strcpy(variable_parts[0], "Ms.");
   strcpy(variable_parts[1], "Sally");
   strcpy(variable_parts[2], "Smith");
   strcpy(variable_parts[3], "123 Main Street");
   strcpy(variable_parts[4], "Anytown");
   strcpy(variable_parts[5], "NY");
   strcpy(variable_parts[6], "12345");

   char letter_template[] = "To: 0 1 2\n3\n4, 5 6\n\nDear 0 2: \n\n"
      "The 2 family may be the lucky winner in the C++ sweepstakes.\n"
      "Wouldn't it be exciting if you were the first 4 residents\n"
      "to use ACME's new C++ development environment? etc. etc. \n\n\n";

   vector<char*> fragments = prepare_mailing(letter_template, variable_parts);

   for (int i = 0; i < fragments.size(); i++)
   {
      cout << fragments[i];
   }

   strcpy(variable_parts[0], "Mr.");
   strcpy(variable_parts[1], "Harry");
   strcpy(variable_parts[2], "Morgan");
   strcpy(variable_parts[3], "456 Park Ave");
   strcpy(variable_parts[4], "Everyville");
   strcpy(variable_parts[5], "KS");
   strcpy(variable_parts[6], "67890");

   for (int i = 0; i < fragments.size(); i++)
   {
      cout << fragments[i];
   }

   return 0;
}
```