

Chapter 27 - Typedefs

1. What is a Typedef?

Typedef is a C and C++ statement that allows new data types to be created without the complexity of creating a class. The reasons for using a *typedef* statement are similar to those for using a *struct* statement. Neither can produce a class of objects that come complete with the means of storing data and the functions to manipulate that data and so are of limited use in designing and maintaining large commercial software products. While neither the *typedef* or *struct* are as powerful as a *class* statement, they require far less code to implement. As any programming student knows, not all programs are large and making a *class* in every situation that needs a convenient unit or structure to hold data can produce a diminishing rate of return of results over time expended, i.e. creating a class can take too long for the benefits.

The *typedef* statement has the following form:

```
typedef data-type synonym;
```

where *data-type* is the basic type or class to create a new type from
synonym is the new data-type name; if the new data-type is an array
of the original data-type, the size of the array is place in
brackets at the end of the synonym (*synonym[size]*)

2. Using Typedef

Arrays and functions in C and C++ have certain limitations. Arrays of null terminated strings are hard to create. Multidimensional arrays are hard to pass or return in functions. While multidimensional arrays will be studied shortly, an array of null terminated strings could be of use now. Conveniently, using typedefs can help us overcome this limitation.

Here are some examples of *typedefs* of convenience:

```
typedef char string15[15]; // a type called string15 that is a string of 15
                          // characters is created
```

```
typedef char line[100];   // a type called line that is a string of 100
                          // characters is created
```

Taking these example types, convenient arrays of strings can be easily created. For example:

```
string15 name[100];      // an array called name is created consisting of 100
                          // 15 character strings; this array can be passed as a
                          // function argument or parameter
line text[50];          // an array called text is created consisting of 50
                          // strings 100 characters long each; this array can
                          // also be passed as a function argument or parameter
```

The following is a simple use of an array of null terminated strings using the typedef that created the data-type `string15`:

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

typedef char string15[15];

void input(string15 n[ ]) {
    for (int x=0; x<5; x++) {
        cout << "Enter Name: ";
        cin >> n[x];
    }
}

void output(string15 m[ ]) {
    for (int y=0; y<5; y++)
        cout << m[y] << endl;
}

void main( ) {
    string15 names[5];
    input(names);
    output(names);
}
```

In this example, not only is `string15` used to create an array of strings, `string15` is used as the data type when passing the array of strings to two functions. As with any array, arrays of type `string15` are passed by reference.

Programming Assignment 27.1

Create and run the above example program.

Programming Assignment 27.2

Alter the program of 27.1 so that the data is written to a file in a third function with called `outputToFile`.

Programming Assignment 27.3

The following program is a simple example of using an array of strings to receive text data from a file, which is then displayed on the screen:

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

typedef char line[100];

void main() {
    line text[50];

    ifstream infile;
    infile.open("c:\\junk\\data.txt");
    if (!infile){
        cout << "No such file!" << endl;
        return;
    }
    int x = 0;
    do {
        infile.getline(text[x], 100, '\n');
        x++;
    } while (!infile.eof() && x<49);

    for (int y=0; y<x; y++)
        cout << text[y]<< endl;
}
```

Create and run the above example program. Change the file path to one that points to a text file on your system.

Programming Assignment 27.4

Alter the program of 27.3 so that the data is read from the keyboard and stored in the array. When the user enters a period as the first item on a line or has entered the maximum number of lines, ask the user if the data is to be saved. If it is, get a file path from the user and write the data to that path.

Programming Assignment 27.5

Alter the program of 27.3 so that the file path is entered by the user into a variable of type string and the array type is not given defined in the typedef statement but is of type string. Open whatever file for input that the user enters. To do this, you will have to convert the string variable that contains the path to the input file to a null-terminated string (C-string) with the string class member function `c_str()`. For example:

```
string fileName;  
:  
:  
:  
inFile.open(fileName.c_str( ));
```