C / C++ PROGRAMMING

Program Organization and Comments

Dan McElroy

Programming Organization

The layout of a program should be fairly straight forward and simple. Although it may just look like a bunch of funny letters and symbols, once you recognize the pattern of how things are organized it makes it much easier to understand what is happening in the program.

Program Organization

The title block at the top of the program is made up of comments that briefly describe the program, date, version, programmer, etc.

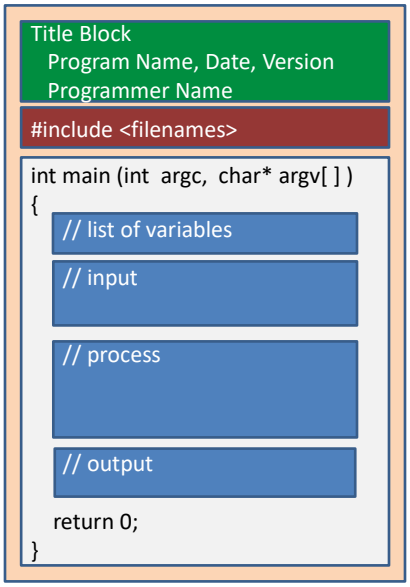
The **include** files come next.

The name of the program section, in this case it is **main**, and its "argument" list.

Curly braces { and } surround the block of code that belongs to main.

The body of the program is organized into the list of variables, input, processing and output.

A return statement ends the program.



Comments

C and C++ Style Comments /* --- */ and //-------

Title block

File name, list of functions/subroutines and their arguments, programmer name, date and revision

Variable descriptions

Description for blocks of code or data

Comments on lines

Reasons for Comments

Comments are written in a human language to help you or the next person better understand what the computer language is doing. Put good comments in your code even if you think that no one else will ever see them. There have been times that I was deeply involved in a project and knew every part of the project and did not think that I needed to comment my code because I knew it so well. I then started a new project and later returned to my first project and had to spend an extra amount of time and effort to figure out what I had done in the old project because I did not have good comments.

The TITLE Comment Block

1) Use a comment block at the top of each file to indicate the program's name, the programmer's name, the date and version of the program, and a brief description of the program and any input or output parameters. The helps identify what the file or program is used for without needing to read the program code and try and figure out what it does.

```
// AvgTemp.cpp : Compute the average temperature
// Version : 1.0
// Date:  1/27/2014
// Programmer: Dan McElroy
// Class: CIS054 C/C++ Programming
//
// Inputs:  10 Temperatures
// Output:  Average Temperature
```

Comment Each Paragraph

2) English essays either use a blank line between paragraphs or indent each paragraph with a few spaces at the beginning of the paragraph. When you use several lines of code that are related to each other and do a specific task, put a blank line and a comment to briefly describe the code.

```
// Compute RegHours and OvertimeHours
if (Hours <= 40)
{
        RegHours = Hours;           // only regular hours
        OvertimeHours = 0.0;        // no overtime
}
else
{
        RegHours = 40;              // regular pay first 40 hours
        OvertimeHours = Hours-40;   // OT for anything over 40
}

// Compute the paycheck
RegPay = RegHours * PayRate;
OvertimePay = OvertimeHours * PayRate * 1.5;
Paycheck = RegPay + OvertimePay;
```

Comment Tricky Code

3) Put a comment on individual lines of code if you did anything that may take some extra work figuring out what the code is doing.

```
// Compute RegHours and OvertimeHours
if (Hours <= 40)
{
        RegHours = Hours;           // only regular hours
        OvertimeHours = 0.0;        // no overtime
}
else
{
        RegHours = 40;              // regular pay first 40 hours
        OvertimeHours = Hours-40;   // OT for anything over 40
}

// Compute the paycheck
RegPay = RegHours * PayRate;
OvertimePay = OvertimeHours * PayRate * 1.5;
Paycheck = RegPay + OvertimePay;
```

C and C++ Style Comments

C++ comments

```
// Project Name:  dogYears (C++ project)
// Description:   Convert dog years to human
// Programmer:    Dan McElroy
// Date:          October 9, 2013


#include <iostream>   // for cin
using namespace std;

int  main()
{
    // define and initialize the
    double age = 0;
    double dogYears  = 0;
```

C comments

```
/*  Project Name:  dogYears (C-project)
    Description:   Convert dog years to human
    Programmer:    Dan McElroy
    Date:          October 9, 2013
*/

#include <stdio.h>  /* for printf and scanf */

int  main()
{
    /* define and initialize the variables */
    double age = 0;
```

1. The Visual Studio editor puts comments in green
2. Most modern compilers will accept both C++ style comments and C style comments
3. C++ comments start with a double slash **//** and stop at the end of line
4. C-style comments start with **/*** and end with ***/**

Header Files - #include

The #include statement is used to merge code from another file into your program as the program is being compiled. Header files get their name because the #include statement is placed at the top of the program.

Although you might not believe it, C and C++ make programming easier by providing header files. The header files provide information to the compiler needed to compile the rest of your program.

Header Files

The compiler needs to know the data type for each input parameter and return value for every function/procedure that will be used in your program, and for each function/procedure that they may in turn reference. Instead of you needing to declare all of these things, they are placed in files located in the compiler's header sub-folder.

All you need to do is provide the name of the header file as part of a #include statement. Easy?

Header Files

C-language

scanf and **printf** provide formatted input and output routines. To use these routines, put #include <stdio.h> at the top of your program. Note: stdio is pronounced "standard I/O". To use the math functions such as sin, cos, tan, sqrt, use **#include <math.h>**

C++ language

cin and **cout** are used for input and output. To use these routines, put #include <iostream> at the top of your program. To use the math functions such as sin, cos, tan, sqrt, use **#include <cmath>**
Note: C++ includes **cmath** not **math.h** The **math** header file for C++ does a few things of its own, then internally includes **math.h**

Header Files

When to use angle brackets or quotes

Use the **angle brackets** <…> around the file name if the header file is part of the compiler and located in the compiler's header sub-folder.

Example: #include <math.h>

Use **quotes** "…" around the file name if the header file is located in a sub-folder that is part of your project. These would typically be header files that you have written

Example: #include "mystuff.h"

Header Files - #include

C++

```
// Project Name:   dogYears (C++ project)
// Description:    Convert dog years to human
// Programmer:     Dan McElroy
// Date:           October 9, 2013

#include <iostream>   // for cin
using namespace std;

int  main()
{
    // define and initialize the
    doubl
    doubl
```

C and C++

```
// Project Name:   dogYears (C-project)
// Description:    Convert dog years to human
// Programmer:     Dan McElroy
// Date:           October 9, 2013

#include <stdio.h>  /* for printf and scanf */

int  main()
{
                                  iables */
```

1. Header files are merged into your program when it is compiled
2. Header files located in the compiler's directory are identified by the angle brackets **<filename>**
3. Header files located in your program's directory are identified by double quotes **"filename"**

Header Files - #include

C++

```
// Project Name:  dogYears (C++ project)
// Description:   Convert dog years to human
// Programmer:    Dan McElroy
// Date:          October 9, 201

#include "stdafx.h"    // for Micr
#include <iostream>    // for cin
using namespace std;

int  main()
{
    // define and initialize the
    doubl
    doubl
```

C and C++

```
// Project Name:  dogYears (C-project)
// Description:   Convert dog years to human
// Programmer:    Dan McElroy
// Date:          October 9, 2013

#include <stdio.h>  /* for printf and scanf */

int  main()
{
                                    iables */
```

1. The C++ console in (**cin**) and console out (**cout**) need iostream
2. The C-console in (**scanf**) and console out (**printf**) need stdio.h
   stdio is pronounced Standard-I-O
3. The **namespace** feature in C++ is provided to make it easier to keep parts of large programs separate from each other
4. There **is** a semicolon **;** at the end of **using namespace std;** but **not** at the end of the #include statements.

int main (…

The **`int main(…`** statement is used to start all C and C++ console application programs. There are several ways that the **`int main(…`** statement can be written. In each case, the only space that is required is after the keywords **`int`** or **`void`**.

```
int main( )
int main(int argc, char* argv[])
```

Microsoft version of int main(

Microsoft likes to make their stuff incompatible with everyone else, so they start Visual Studio C and C++ programs

```
int _tmain(int argc, _TCHAR* argv[])
```

and then they convert it back to normal before compiling.

If you are using Microsoft, you can leave the int main(the way they defined it, or you can use the standard version of int main. Both work fine.

int main parameter list

When declaring a function such as main, an optional set of parameters can be placed within the parentheses ( ). This permits another part of the program to pass information to the function.

`main` is usually declared with the **`int argc`** and **`char* argv[]`** parameters inside the parentheses, but these parameters do not need to be supplied if they are not used inside **`main`**.

`int argc` identifies the number of arguments passed

`char* argv[ ]` provides an array of pointers to the character strings supplied when the program was started.

More detail will be provided when these parameters are used in a program.

int main or void main

Placing a data type in front of a function declaration indicates the type of data passed back by the return statement to a higher level program or program routine.

The C and C++ console programs are called/launched by the operating system and have the ability to return a single integer indicating the exit status of the program.

A console program that starts with **`int main`** should have a **`return 0;`** statement at the end of the program to indicate successful completion of the program. A program that starts with **`void main`** can have a **`return;`** statement, but it can not return an integer (like 0).

List of Variables and Constants

Variables in C and C++ are declared as follows and can be initialized at the same time they are declared:

data_type variable_name semicolon

Examples:

```
int  length;
int  count = 20;
double price;
```

In C++, constants are declared with the keyword **`const`** before the data type. In C, use the #define with no = or ;
The value of a constant can not be changed after it is defined. For example to define a tax rate of 8.75%

```
const double TAX_RATE = 0.0875;  // 8.75% C++
#define TAX_RATE 0.0875  // 8.75% C
```

Input Data to be Processed

Data must be input into the program before it can be processed. Data can be input from the keyboard or another device such as a disk file.

When data is input from the keyboard, it is necessary to provide a 'prompt' message on the screen that asks for the data, otherwise the user may just see a flashing cursor _ and not know that they need to do anything, or what type of data should be input.

The Prompt Message

The word "prompt" comes from the theater. A person might hold up a card to help the actor remember what line he or she needs to say next. TV shows use a machine called a teleprompter.

C++

```
cout << "Hours worked: ";  // prompt
cin  >>  hours;            // input the hours
```

C

```
printf ("Hours worked: "); /* prompt */
scanf  ("%lf", &hours);    /* input the hours */
```

Provide a colon : and a space at the end of the prompt to keep the user's input from sitting right next to the prompt message. If using the C language, don't forget the ampersand & inside of scanf.

Process the Data

The processing of the data will be dependent on the requirements of the program.

You may need to create variables in addition to the variables needed for the input and output data.

Output the Result

Outputting a text string in C or C++ is fairly easy.

```
cout << "You passed the test";  // C++
printf ("You passed the test"); /* C */
```

or

```
char message[] = "You passed the test";
cout << message;   // C++
printf ("%s", message);  /* C */
```

The "%s" in the **printf** statement is part of the format string that indicates that a string will be output. In this case, the string to be output is the contents of the variable **message**.

cout – Formatted Output

cout will display a floating point number (double) any way it wants. It could show up with no digits past the decimal point, a whole lot of digits, or the display may be in scientific notation. C++ provides several routines that are part of the I/O manipulation package. You need to use **#include <iomanip>** at the top of your program to use these routines.

cout – Formatted Output

Sample of using cout to display a double with two decimal places:

```
#include <iostream>
#include <iomanip>
using namespace std;
   .
   .
   .
double Paycheck = 183.75;
cout << setiosflags(ios::fixed);
cout << setiosflags(ios::showpoint);
cout << setprecision(2) << Paycheck << endl;
```

The **endl** represents end-of-line and is used to move the cursor to the next line on the screen. Make sure that you type the last character of endl as a small 'L' and not a '1'.

printf – Formatted Output

printf can be much easier to than the **cout** routines, but the full documentation on printf is several pages long. The example below shows only how to use **printf** to display a double with two decimal places.

```
#include <stdio.h>
   .
   .
   .
double Paycheck = 183.75;
printf ("%.2lf\n", Paycheck);
```

The lower-case letters LF indicate that a long-float value is supplied. A long-float is a double. In this case the value is supplied by the contents of **Paycheck**. The \n represents end-of-line and is used to move the cursor to the next line on the screen.