Program Control Structures

Michael Griffiths
Corporate Information and Computing Services
The University of Sheffield
Email m.griffiths@sheffield.ac.uk

Control

- Sequence Structures
- Selection Structures
  - if… else statements
  - switch structures
- Repetition Structures
  - for loops
  - while loops

Conditional Statements Using if…else

- The if statement allows decision making functionality to be added to applications.
- General form of the if statement is:

```
if(condition)
statement;
```

The University Of Sheffield.

The University of Sheffield Research IT

Conditional Operators

- Compare values using conditional operators.
- ==   equal to
- >   greater than
- <   less than
- >=   greater than or equal to
- <=   less than or equal to

Using else

- An alternative form of the if statement is

```
if(condition)
        statement;
else
        statement;
```

If the condition is true the first statement is executed if it is false the second statement is executed.

Simple if Example Demonstrating Syntax

```
int main(void)
{
   float f1;
   printf("Enter a floating point number.\n");
   scanf("%f",&f1);
   if(f1<0)
     printf("Please enter a value greater 0\n");
   else if (f1>100)
     printf("f1 is greater than 100\n");
   else
     printf("The value is %f\n",f1);

   return 0;
}
```

Demonstration

- Build and run the example if1.c and if2.c
  - Note the condition operator in round brackets
  - Use of {} when multiple statements are used
  - Test the program with different conditions
- Build and run if3.c
  - What is wrong with this code?
  - Was there are compiler warning?
- Build and run the programs ifelse.c and ifelseifelse.c

Multiple selection Structures Using Switch

- Used for testing variable separately and selecting a different action

```
switch(file)
{
case 'm': case 'M':
            ++nMaxima;
break;
case 't': case 'T':
            ++nTitania;
break;
default:            /*Catch all other characters*/
            ++nOther;
break;
} /*End of file check switch */
```

Demonstration

- Build and run the example switch1.c
  - What happens when the break statements are commented out (put a // at the start of the line)
- switch2.c is a menu application
  - The program is case sensitive
  - Modify the program so that the user can type either upper or lower case characters

Repetition Using while

- Execute commands until the conditions enclosed by the while statement return false.

```
while(conditions)
{
statements;
}
```

while

- Good practice to always use {} in a do while loop

```
while(conditions)
{
        statements…;
Statements…;
}
```

do … while

- Good practice to always use {} in a do while loop

```
do
{
          statements…;
Statements…;
}
while(conditions)
```

Demonstration

- Build and run the example while1.c
  - Modify the loop so that it counts to 20
  - Modify the loop so that it counts to 20 in steps of 2
- Build and run dowhile.c

While example demonstrating a countdown

```
int main (void)
{
  int n;
  printf( "Enter the starting number\n");
  scanf("%d".&n);
  while (n>0) {
    printf("%d, ",n);
    --n;
  }
  printf("FIRE!\n");
  return 0;
}
```

Try this code then modify it by introducing a second variable m initialised to 10. Use the && operator to add a test m<10 to the Condition in the while statement.

Example of while and if statement usage

```
while(files<=5)
 {
printf("Enter file location(1=Titania, 2=Maxima): ");
        scanf("%d", &result);

if(result==1)
                ntitania_files = ntitania_files+1;
        else if(result==2)
                nmaxima_files = nmaxima_files+1;
        else
                nother_files=nother_files+1;

        files++;
 }/*End of while file processing loop*/
```

Continue counting until Maximum number of files entered (5)

Request and get user input

Use conditions to update variables

Increment counter

Counter Controlled Repetition

- Components of a typical for loop structure

```
for(expression1; expression2; expression3)
statement;
```

example

```
for(counter=1; counter<=10, counter++)
statement;
```

Demonstration

- Build and run the example for1.c
- Build and run nestedfor1.c

for loop example

- Example program for.c
  - Modify the program so that it performs a count down

```
main()
{
       int counter, nsteps=10;
        /* initialisation, repetition condition and increment */
        /* are all included in the for structure header       */
        for(counter=1; counter<=nsteps; counter++)
                             printf("%d\n", counter);
       return 0;
}
```

Practical Examples – basic coding

- Inspect, Compile and run the following
- Finding a root using the Newton-Raphson method
  - While statement
- Finding a root by method of bisection
  - If statement, while statement
  - And simple one line function!

Compilers

Language	GNU	Portland	Intel
C	gcc	pgcc	icc
C++	g++	pgCC	icpc
Fortran 77	g77	pgf77	
Fortran90/95	gfortran	pgf90	ifort

Invoking the Compiler

- Compiling FORTRAN Programs
  - g77 –o mycode [options] mycode.f
- Compiling c/c++ Programs
  - gcc –o mycode [options] mycode.c

Compiler Options

Option	Action
-s	remove any symbol and object relocation information from the program. This is used to reduce the size of program and runtime overhead
-c	Compile, do not link.
-o exefile	Specifies a name for the resulting executable.
-g	Produce debugging information (no optimization).
-llibrary_name (lower case L)	Link the given library into the program. e.g. include math library by using option -lm
-Idirectory-name (upper case I)	Add directory to search path for include files
-O N	Set optimisation level to N
-Dmacro[=defn]	Define a macro

Summary

- Program Structure
- Control Structures for C Programming
- Compilers and development Environments