

C Programming

Learning Package 3

Conditional Expressions and Decisions

Dr Scott Turner

School of Science and Technology

University of Northampton



**Introduction**

In this Learning Package we are going to look at how the flow of a program can be altered depending on tests within the program.

**Learning outcomes**

At the end of this Learning Package, the reader should be able to:

* Describe what a condition statement is.
* State the possible logical operators used in C.
* Demonstrate the ability to write programs that allow different outcomes, which depend on the value of a certain condition.

**Study guide**

Session 1:

* 1. Conditional expressions SAQ 3.1

Session 2:

* 1. The If-statements Task 3.1

Session 3:

* 1. Switch statements Task 3.2

Session 4:

Exercises

* 1. **Conditional expressions**

Read from page 84 to the end of section labelled *4.3 Logical operators* on page 89 of the module textbook.

SAQ 3.1

Fill in the missing words

1. Any valid C **e n** can be used as a **t t** in a **c\_ \_l e\_ n**. The value of a **c l e\_ \_n** will often be

**d d** by **c\_ g** values using **r l o\_ rs** (e.g. >, <,

==). A **r l o rs** is used to **c \_e t\_o e\_ \_ns**

and produce an integer value 1 if it is **t\_\_e** and 0 if it is **f e**.

1. As well as relational operators there are **l \_l o rs**. There are three of these, (&&), (||) and (!).
	1. **The If-statement**

Read from the section labelled *4.4 The If Statement* on page 89 to the end of the section labelled *4.5 The If…Else Statement* on page 96 of the module textbook. The various forms of an if-statement do not cause the program to repeat earlier instructions.

Figure 3.1 Flow diagram for an if-statement.

Figure 3.1 shows how the program flow (the order instructions are carried out) for an if-statement. In this diagram a test is made (the diamond) if the results of the test was true then statement 1 is carried out, and then statement 2 is carried out. If the test was false then statement 1 is not carried out, and the program moves onto statement 2. Figure 3.2 shows the flow through the program for an if-else statement, it shows if the decision is true then the statement 1 is performed; else if the decision is false then do statement 2. Figure 3.3 shows the flow through the program for an if- else if - else statement.

* + - If decision 1 is true then do statement 1 and the program jumps out of this block of code (it does not perform statement 2 or 3).
			* If decision 1 is false do then test decision 2.
			* If decision 2 is true then do statement 2,
		- if nether decision 1 or decision 2 are true then do statement 3.

Figure 3.2 Flow diagram for an if-else statement

Figure 3.3 Flow diagram for an if-else if–else statement

Notice in all of three figures the direction of the arrows never goes up the page; this is because these statements never make the computer go directly to an instruction that has already been carried out earlier. We will look at instructions that do this in Learning Package 4. The various if-statements alter which is the next block of code further down the program is performed next.

Task 3.1

Type in, compile and execute program 4.1 (Boring to type in though).

SAQ 3.2

Many of the condition statements in program 4.1 are of the form:

if (twenty>0) {

printf("\t %d twenty pence coin",twenty); if (twenty>1)

printf("s");

printf("\n");

}

However, there also lines like

if (five>0)

printf("\t %d five pence coin",five);

Why does one need a nested if (an if statement inside another if-statement) and the other does not?

* 1. **Switch statements**

Read the section labelled *4.6 The Switch statement* on pages 96-100 of the module textbook.

SAQ 3.3

Filling the missing words

If the *integral\_expression* in the switch statement **m s** the value of the

**c\_ t** in the **c e** statement then **c\_ l** is passed to the **s t i\_ \_y** following that case label. If no **m\_ s** occur and a default case label is present then **c \_l** is passed to that **s\_ t**.

Task 3.2

Type in, compile, and execute program 4.3

**Exercises**

1. Do exercises starting on page 101 of the module textbook do 1.
2. Do exercise 3, one using if statements and a second version of the same problem using switch statements.

(c) Do exercise 5 (Remember M=1000, D=500, C=100, L=50, X=10, V=five,I=1, IV=4, VI=6, IX=9).

**Answers**

SAQ 3.1

Fill in the missing words

1. Any valid C **expression** can be used as a **test** in a **conditional expression**. The value of a **conditional expression** will often be **determined** by **comparing** values using **relational operators** (e.g. >, <, ==). A **relational operator** is used to **compare two expressions** and produce an integer value 1 if it is **true** and 0 if it is **false**.
2. As well as relational operators, there are **logical operators**. There are three of these, **AND** (&&), **OR** (||) and **NOT**(!).

SAQ 3.2

This does not have anything to do with C it is a logical problem. The if-statement that is nested inside the other if, is there to add an s to the end of coin or pound to make the expression plural. Therefore, there will never be more than 1 stored in the variable five.

Imagine what we have as change is 45 pence because they way the program works we end up with twenty=2 and five=1. So we need to say we that more than one twenty pence coin is necessary.

SAQ 3.3

If the *integral\_expression* in the switch statement **matches,** the value of the **constant** in the **case** statement then **control** is passed to the **statement immediately** following that case label. If **no matches** occur and a default case label is present then **control** is passed to that **statement**.

**Selected exercises**

1. Example

#include <stdio.h> void main()

{

int value\_1,value\_2,res1; char op;

scanf("%c%d%d",&op,&value\_1,&value\_2); if (op=='a')

{

res1=value\_1+value\_2;

}

else if (op=='s')

{

res1=value\_1-value\_2;

}

else if (op=='m')

{

res1=value\_1\*value\_2;

}

else if (op=='d')

{

}

else

{

}

res1=value\_1/value\_2;

res1=-1;

printf("\nResult of %d and %d is %d\n",value\_1,value\_2,res1);

}

1. An example solution

#include <stdio.h> void main()

{

int year,rest,temp; printf("\n Enter Year: "); scanf("%d",&year);

printf("\n\n"); temp=year/1000; switch(temp)

{

case 1:printf("M");break; case 2:printf("MM");break; case 3:printf("MMM");break; case 4:printf("MMMM");break; case 5:printf("MMMMM");break; case 6:printf("MMMMMM");break; case 7:printf("MMMMMMM");break; case 8:printf("MMMMMMMM");break; case 9:printf("MMMMMMMMM");break;

}

rest=year-(temp\*1000); temp=rest/100; switch(temp)

{

case 1:printf("C");break; case 2:printf("CC");break; case 3:printf("CCC");break; case 4:printf("CD");break; case 5:printf("D");break; case 6:printf("DC");break; case 7:printf("DCC");break; case 8:printf("CCM");break; case 9:printf("CM");break;

}

rest=rest-(temp\*100); temp=rest/10; switch(temp)

{

case 1:printf("X");break; case 2:printf("XX");break; case 3:printf("XXX");break; case 4:printf("XL");break; case 5:printf("L");break; case 6:printf("LX");break; case 7:printf("LXX");break; case 8:printf("XXC");break; case 9:printf("XC");break;

}

rest=rest-(temp\*10); temp=rest; switch(temp)

{

case 1:printf("I");break; case 2:printf("II");break; case 3:printf("III");break; case 4:printf("IV");break; case 5:printf("V");break; case 6:printf("VI");break; case 7:printf("VII");break; case 8:printf("IIX");break; case 9:printf("IX");break;

}

}