

C Programming

Learning Package 4

Going Loopy

Dr Scott Turner

School of Science and Technology

University of Northampton



**Introduction**

In Learning Package 3 we looked at how to alter the flow of a program depending on a conditional statement (a test). In this Learning Package, we are going to look at how to make the program repeat blocks of code as many times as is needed.

**Learning outcomes**

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

* State three ways to make a program repeat blocks of code.
* Discuss how conditional statements are used to enable the blocks of code to repeat.
* Discuss the difference between these loops and the branching statements in Learning Package 3.
* Implement programs using loops

**Study guide**

Session 1:

Section 4.1

SAQ 4.1

SAQ 4.2

Task 4.1

Section 4.2

Task 4.2

Session 2:

Section 4.3

Task 4.3

Session 3:

Section 4.4

Task 4.4

Session 4:

Exercises

* 1. **While**

Read from page 103 to the end of the section labelled 5.2 *The While loop* on page 109 of the module textbook.

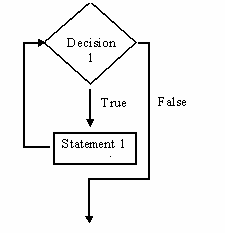


Figure 4.1 While loop

SAQ 4.1

For each of the following blocks of code state for what values of x will the blocks of code ‘loop’:

1. while(x>3)

{

printf("It worked!");

}

1. while(x==3)

{

printf("It worked!");

}

1. while(x!=3)

{

printf("It worked!");

}

SAQ 4.2

For each of these programs (without typing them in) describe what you think will happen when these programs are run.

1. #include <stdio.h> void main()

{

int x; x=6;

while(x>3)

{

printf("It worked!");

}

}

1. #include <stdio.h> void main()

{

int x; x=7;

while(x>3)

{

printf("\nIt worked!"); x=x-1;

}

}

Task 4.1

1. Type in, compile, and execute the programs in SAQ 4.2.
2. Type in, compile, and execute program 5.1 on page 108 of the module textbook.
3. For all the programs in (a) and (b) put comments into the program code to describe the programs operation.
   1. **Do-while loop**

Read the Do … While loop section from page 109 to 111 of the module textbook.

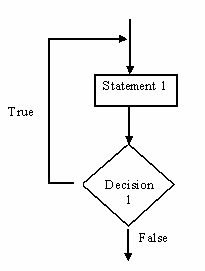


Figure 4.2 Do-While loop

Task 4.2

Type in, compile, and execute the following programs, what do they do?

1. #include <stdio.h> void main()

{

int x; x=7;

do

{

printf("\nIt worked!"); x=x-1;

}while(x>3)

}

1. #include <stdio.h> void main()

{

int value\_1; do

{

printf("\nEnter 1 or 2 to stay in the loop"); scanf("%d",&value\_1);

}while(value\_1<=2); printf("\n\nOut of the loop\n\n");

}

1. (i) For (b) can you find values other than 1 or 2 that will make the code in the loop repeat.

(ii) Alter the program in (b) above so that ONLY when 1 or 2 are entered will the code in the loop repeat.

* 1. **For loop**

Read from the beginning of the section labelled *5.4 the for loop* on page 111 to the end of the section labelled *5.5 The comma operator* on page 117 of the module textbook.

Task 4.3

Compile and execute the following programs, and describe what they do by commenting the programs:

1. #include <stdio.h> void main()

{

int value\_1;

for (value\_1=0;value\_1<10;value\_1=value\_1+1)

{

printf("\n Value\_1=%d \n",value\_1);

}

}

1. #include <stdio.h> void main()

{

int value\_1, value\_2;

for (value\_1=1;value\_1<8;value\_1=value\_1+2)

{

printf("\nValue\_1=%d Value\_2=",value\_1); for (value\_2=1;value\_2<=8;value\_2=value\_2+2)

{

printf("\t%d",value\_2);

}

}

printf("\n");

}

(b) program 5.2 on page 116

* 1. **Miscellaneous**

Read sections *5.6 Example-Prime Numbers* and *5.7 Arrays - A Quick look* of the module textbook

Task 4.4

(a) Compile and execute program 5.3 on page 121.

**Exercises**

1. Modify program 5.1 so that it counts upper and lower case vowels.
2. Do Exercise starting on page 122 of the module textbook, do 1, 3, 4, 6(b), 7, 8, 9(a, b).

**Answers** SAQ 4.1

For each of the following blocks of code state for what values of x will the blocks of code loop:

1. When the value of x is greater than 3, so 3 would not make the program loop.
2. Only when x is exactly 3.
3. Any value of x other than 3.

SAQ 4.2

For each of these programs (without typing them in) describe what you think will happen when these programs are run.

1. This program just keeps printing **It worked!** on the screen. This is because the value of x is always 6 and the program loops when x is greater than 3 because of the while statement. The programming will continue repeating (looping) while x is greater than 3 and x is stuck at 6 so it continues repeating. This is an infinite loop.
2. This program prints **It worked!** on the screen four times. Initially x equals 7, 7 is greater than 3 so the loop begins. The message is displayed on the screen. There is a second statement in this repeating block of code which decreases x by 1 each time. So x is now 6, and the loop continues (6>3) but each time x decreases by 1. Eventually x will become equal to 3 and when x is tested to see if it is greater than 3, it is not so the program ignores what is in the loop.

**Selected tasks**

4.2 (c) (i) Values less than 1, e.g. 0, -1,-2, etc will also make the code in the loop of 4.2(b) repeat.

(ii) Program below will only loop when value\_1 is either 1 or 2.

#include <stdio.h> void main()

{

int value\_1; do

{

printf("\nEnter 1 or 2 to stay in the loop"); scanf("%d",&value\_1);

}while(value\_1==2||value\_1==1); printf("\n\nOut of the loop\n\n");

}

**Selected exercise**

1. #include <stdio.h>

#define TRUE 1

#define FALSE 0

main()

{

char ch,last\_ch,more=TRUE;

int a\_count=0, e\_count=0,i\_count=0,

o\_count=0,u\_count=0;

printf("\nPlease enter test (as much as you like)\n"); printf("Indicate the end with two successive returns\n"); ch=getchar();

while (more) {

if (ch=='a'||ch=='A') a\_count++;

if (ch=='e'||ch=='E') e\_count++;

if (ch=='i'||ch=='I') i\_count++;

if (ch=='o'||ch=='O') o\_count++;

if (ch=='u'||ch=='U') u\_count++;

last\_ch=ch; ch=getchar();

if (ch=='\n' && last\_ch=='\n') more=FALSE;

}

printf("The VOWEL COUNT PROGRAM");

printf("\n\t\ta\te\ti\to\tu\n"); printf("number found:\t"); printf("%d\t%d\t%d\t%d\t%d\n",

a\_count,e\_count,i\_count,o\_count,u\_count);

}

1. (3) and (4) in this case the do-while means that only one ch=getchar() line needs to be used.

#include <stdio.h>

#define TRUE 1

#define FALSE 0 main()

{

char ch=' ',last\_ch,more=TRUE;

int a\_count=0, e\_count=0,i\_count=0, o\_count=0,u\_count=0;

printf("\nPlease enter test (as much as you like)\n"); printf("Indicate the end with two successive returns\n"); do{

last\_ch=ch; ch=getchar(); switch(ch) { case 'a':

a\_count++;break; case 'e':

e\_count++;break; case 'i':

i\_count++;break; case 'o':

o\_count++;break; case 'u':

u\_count++;break;

}

if (ch=='\n' && last\_ch=='\n') more=FALSE;

}while(more);

printf("The VOWEL COUNT PROGRAM");

printf("\n\t\ta\te\ti\to\tu\n"); printf("number found:\t"); printf("%d\t%d\t%d\t%d\t%d\n",

a\_count,e\_count,i\_count,o\_count,u\_count);

}

(b)(8)

#include <stdio.h> main()

{

int start=1,step=1,end=20,value2,loop; float recip;

printf("\nEnter the start, end and step size: "); scanf("%d %d %d", &start,&end,&step);

if (start<0)

printf("\n\nStarting point is negative"); else if (end<start)

printf("\nStart value is greater than End value"); else if (step>(end-start))

printf("\nOut of range");

else{

printf("\n\tinteger\tsquared\treciprocal\n"); for (loop=start;loop<=end;loop=loop+step)

{

value2=loop\*loop; if (loop!=0)

{

}

else

{

}

}

}

}

recip=1./loop; printf("\t%d\t%d\t%f\n",loop,value2,recip);

printf("\n\nDivide by zero error\n"); printf("\n\t%d\t%d\n",loop,value2);

(b)(9)

#include <stdio.h>

main()

{

int terms=1,loop;

double x, squared,fact,sum,sign1, past,error,error2,error3,value1;

x=3.1421/2;

error3=0.001\*0.001; squared=x; sign1=1; fact=1; sum=squared; loop=3;

do

{

past=sum; squared=squared\*x\*x; fact=fact\*loop\*(loop-1); sign1=sign1\*(-1); sum=sum+(sign1\*(squared/fact)); loop=loop+2;

error=past-sum; error2=error\*error;

printf("past=%f\tsum=%f\terror=%f\n",past,sum,error); terms++;

}while(error2>error3);

printf("\nsin(%f)=%f \tNumber of terms=%d\n",x,sum,terms-1);

}