

C Programming

Learning Package 5

Functions

Dr Scott Turner

School of Science and Technology

University of Northampton



**Introduction**

In this Learning Package we are going to look at the concepts of a function; also how and why functions are used within C programs.

**Learning outcomes**

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

* State what is a function.
* We able to implement functions that
	+ Return a value
	+ Pass values into a function
* Implement prototyping of functions.

**Study guide**

Session 1:

Section 5.1

SAQ 5.1

Task 5.1

SAQ 5.2

Session 2:

Section 5.2

SAQ 5.3

Task 5.2

Task 5.3

Session 3:

Section 5.3

SAQ 5.4

Section 5.4

SAQ 5.5

Task 5.4

Session 4:

Exercises

* 1. **Introduction to functions**

Read from page 125 to the end of the section labelled *5.3 Function Types other than int* on page 132 of the module textbook.

SAQ 5.1

Fill in the missing words.

1. The use of *return* **t s c l** of the **p m b k** to where the function was **c d**, it is also used to pass back to where the function was called, a **v e.**
2. A **f n** is called by **r ring** to its **n e** followed by

**b ts** enclosing any **p rs** that are required.

1. When a function **c l** is met **c l** of the **p m** is passed to the **c e** for the function, when it is completed, the program carries on from the point in the program where the function call occurred.

SAQ 5.2

State whether the following line contains a call to a function:

1. printf("hello");
2. int f;

(c) a=1+2;

(d) add(1,2);

(e) a=add(3,4);

Task 5.1

1. Type in and try out the program shown below. What does it do?

#include <stdio.h> void ch\_test(void)

{

char ch;

printf("\n Please enter a character: "); ch=getchar();

if (ch>='a' && ch<='z'){

printf("\n a lower case latter was entered\n"); return;

}

if (ch>='A' && ch<='Z'){

printf("\n an upper case letter was entered\n"); return;

}

if (ch>='0' && ch<='9'){

printf("\n a digit was entered\n"); return;

} else

}

printf("\n you entered some other character\n");

void main()

{

ch\_test();

}

1. Type in and try out this modified version of program 6.3 on page 126- 127 of the module textbook.

#include <stdio.h> int sum(void)

{

int number, isum=0; scanf("%d",& number); while (number !=0) {

isum +=number; scanf("%d",&number);

}

return(isum);

}

void main()

{

int total;

printf("\n The SUM function \n");

printf("This returns the sum of a number of integers\n"); printf("...to end the series enter 0\n");

total=sum();

printf("\nThe sum of the numbers is %d\n",total);

}

1. Type in and try out program 6.4 on page 130 of the module textbook. What does it do?
2. For each of the programs in this task put comments within the program code to explain what the program is doing.
	1. **Passing data into a function**

Read from the section labelled *6.4 Passing data into a function* starting on page 133 up to the end of section *6.6 Call by Value* starting on page 140 of the module textbook.

SAQ 5.3

For each of the following program from the textbook, state what data type (if any) can be returned from the function and data type (or types) is passed into the function:

1. Program 6.5;
2. Program 6.6;
3. Program 6.8b;
4. Program 6.9;
5. Program 6.10;
6. Program 6.11.

Let’s look at example of a prototyping

#include <stdio.h> int sum(void); void main()

{

int total;

printf("\n The SUM function \n");

printf("This returns the sum of a number of integers\n"); printf("...to end the series enter 0\n");

total=sum();

printf("\nThe sum of the numbers is %d\n",total);

}

int sum(void)

{

int number, isum=0; scanf("%d",& number); while (number !=0) {

isum +=number; scanf("%d",&number);

}

return(isum);

}

This program performs exactly as the program in Task 5.1 (b), the functions main() and sum() are the same. What is different is the position of the definition of function sum() within the program. In task 5.1 (b) it was defined before the function main(). In the program above main() is defined before sum(), but sum() is called from main(). Usually you cannot use something until you have defined it first, so this could be a problem.

The problem is avoided by the line

int sum(void);

The line above says that the function will return values of a certain data type, in this case an integer. It also defines what the inputs to this function are.

This is a prototype, it tells the program that a function of this form exists in the program.

As a second example:

#include <stdio.h>

float sum(float valu\_1); void main()

{

float total,number; printf("\n The SUM function \n");

printf("This returns the sum of a number of integers\n"); printf("...to end the series enter 0\n"); scanf("%f",&number);

total=sum(number);

printf("\nThe sum of the numbers is %f\n",total);

}

float sum(float valu\_1)

{

float isum=0.0; while (valu\_1 !=0.0) {

isum +=valu\_1; scanf("%f",&valu\_1);

}

return(isum);

}

The program above is similar to the previous program except it can handle floating-point numbers and the function sum has a value passed into it. Note that the prototype line float sum(float valu\_1); and the function definition float sum(float valu\_1)are the same expect the prototype ends with a semicolon.

Prototyping means that at the top of your program you have a list of functions within that program. You do not have to use prototyping, if you do not you will have to define all your functions before you define what is in the main() function. The choice is yours.

Task 5.2

1. Combine programs 6.11 to 6.12 from the module textbook into one program.
2. Do the Exercise on page 142 of the module textbook

Task 5.3

1. The program below was designed to take three floating-point values from the program, add them together and display the result. Correct the errors in the program so that it can do this.

#include <stdio.h>

int add\_3(int value\_1, int value\_2)

{

int result result=value\_1+value\_2 return(result)

}

void main()

{

printf('Result=%d /n',add\_3(3,4,5))

}

1. Alter the program above so add\_3() is defined after main(). A prototype of add\_3() is needed.
2. Alter the program so that the program uses three integers entered from the keyboard.
	1. **Functions and headers**

Read section *6.7 Functions and Headers* starting on page 144 of the module textbook.

SAQ 5.4

Fill in the missing words.

A **p m** might use the **l y f ns** getchar() or putchar(), so the **h r f e** stdio.h must be included in the **p m** (By means of

#include <stdio.h>).

* 1. **Storage class**

Read section *6.8 Storage class* starting on page 146 of the module textbook.

SAQ 5.5

Fill in the missing words.

1. The **s e c ses** govern the availability of a **v e** (or

**f ns**) to other **f ns**, parts of the **p m**, or **p ms**.

1. There are four different types of storage class: **a o**, **e n**, **r r** and **s c**.
2. The **a o** storage class is the **d t** class for variables declared

**w hin** a function, and are **l l** to this function. Local means the variables defined in this way have no effect **o e** of the function. The **s e** variable name can be defined in this way in several **f ns** without effecting each other.

1. The **e rn** storage class allows information to be transferred from blocks of codes or from function to function. All variables declared **o e** of a function (including main() ) are by default of this **s e c s**. This storage class is also used to tell the compiler that a variable or function is

**o e** of the present **p m**. To use a set of functions stored in a separate file from the main program, these would have to be declared as **e n**.

1. The **r er** storage class has the same effect as the **a o st e c s** except that compiler is advised to place **v s d d** in this way in *registers*.
2. The **s c** storage class is used when its **v e** is to be **r ned bet n** uses.

Task 5.4

Type in and test Program 6.16 on page 147 of the module textbook. Are any further lines of code needed?

Task 5.5

Put comments into the program shown below, and describe what it does.

#include <stdio.h>

double powx(int a, int b); double MAXNINT;

main()

{

MAXNINT=-1.\*powx(2,(sizeof(int)\*8)-1);

printf("\nMaximum size of negative integer is %f\n",MAXNINT);

}

double powx(int a, int b)

{

int loop;

double result=1;

for (loop=0;loop<b;loop=loop+1)

{

result=result\*a;

}

return(result);

}

**Exercises**

1. Do Exercise starting on page 150-152 of the module textbook, do 1 (code is on page 128 of the textbook), 3 (Hint*: if (scanf("%d",&x)==1) will only be 1 if an integer is entered*), 5, 7, 8.
2. Alter the program you wrote in Task 5.2(b) so that the function adds two floating-point numbers together and then multiplies the result by an integer. The result should be represented accurately.
3. Correct the program below

include <studio.h>; void ch\_test(void); void main()

(

ch\_test1();

)

void ch\_test(int void) (

char ch;

printf("\n Please enter a character: ") ch=getchar();

if (ch>="a" && ch<="z"){

print("\n a lower case latter was entered\n") return;

)

if (ch>="A" && ch<="Z"){

print("\n a upper case letter was entered\n") return;

]

if (ch>="0" && ch<="9")(

print("\n a digit was entered\n") return;

} else

)

print("\n you entered some other character\n")

**Answers**

SAQ 5.1

Fill in the missing words.

1. The use of *return* **transfers control** of the **program back** to where the function was **called**, it is also used to pass back to where the function was called, a **value**.
2. A **function** is called by **referring** to its **name** followed by **brackets**

enclosing any **parameters** that are required.

1. When a function **call** is met **control** of the **program** is passed to the **code** for the function, when it is completed, the program carries on from the point in the program where the function call occurred.

SAQ 5.2

State whether the following line contains a call to a functions:

1. printf("hello") - a call to the function printf()
2. int f; - not a call to a function
3. a=1+2; - not a call to a function
4. add(1,2); - a call to a function add()
5. a=add(3,4); - contains a call to a function.

SAQ 5.3

For each of the following state what data type (if any) can be returned from the function and is passed into the function:

1. character is returned; no values are passed into the function.
2. integer is returned; a character is passed into the function.
3. character is returned; no values are passed into the function
4. integer is returned; a character is passed into the function)
5. integer is returned; no values are passed into the function.
6. integer is returned; three integers are passed into the function.

SAQ 5.4

Fill in the missing words.

A **program** might use the **library functions** getchar() or putchar(), so the

**header file** stdio.h must be included in the **program** (By means of #include

<stdio.h>).

SAQ 5.5

Fill in the missing words.

1. The **storage classes** govern the availability of a **variable** (or

**functions**) to other **functions**, parts of the **program**, or **programs**.

1. There are four different types of storage class: **auto**, **extern**, **register** and **static**.
2. The **auto** storage class is the **default** class for variables declared **within** a function, and are **local** to this function. Local means the variables defined in this way have no effect **outside** of the function. The **same** variable name can be defined in this way in several **functions** without affecting each other.
3. The **extern** storage class allows information to be transferred from blocks of codes or from function to function. All variables declared **outside** of a function (including main() ) are by default of this **storage class**. This storage class is also used to tell the compiler that a variable or function is **outside** of the present **program**. To use a set of functions stored in a separate file from the main program, these would have to be declared as **extern**.
4. The **register** storage class has the same effect as the **auto storage class** except that compiler is advised to place **variables defined** in this way in *registers*.
5. The **static** storage class is used when its **value** is to be **retained between** uses.

**Selected tasks**

Task 5.2(b)

#include <stdio.h>

#define MAXNINT -32768

int maximum(int n, int cmax); main()

{

int i, num,max;

printf("Please enter 10 intgers\n");

printf("...use spaces to seperate the intgers.\n"); printf("and end the list with a return.\n"); max=MAXNINT;

for (i=0;i<10;i++)

{

scanf("%d",&num); max=maximum(num,max);

}

printf("\nThe maximum number entered was %d\n",max);

}

int maximum(int n, int cmax)

{

if (n>cmax)

cmax=n; return cmax;

}

Task 5.3 (a)

#include <stdio.h>

int add\_3(int value\_1, int value\_2, int value\_3)

{

int result; result=value\_1+value\_2+value\_3; return(result);

}

void main()

{

}

(b)

printf("Result=%d \n",add\_3(3,4,5));

#include <stdio.h>

int add\_3(int value\_1, int value\_2, int value\_3); void main()

{

printf("Result=%d \n",add\_3(3,4,5));

}

int add\_3(int value\_1, int value\_2, int value\_3)

{

int result; result=value\_1+value\_2+value\_3; return(result);

}

(c)

#include <stdio.h>

int add\_3(int value\_1, int value\_2, int value\_3); void main()

{

int val\_1,val\_2,val\_3; printf("Enter three integers "); scanf("%d %d %d",&val\_1,&val\_2,&val\_3);

printf("Result=%d \n",add\_3(val\_1,val\_2,val\_3));

}

int add\_3(int value\_1, int value\_2, int value\_3)

{

int result; result=value\_1+value\_2+value\_3; return(result);

}

**Selected exercises**

Exercise (a)

#include <stdio.h> float mean\_1(); main()

{

float m1; m1=mean\_1(5);

printf("\n\nmean=%f",m1);

}

float mean\_1()

{

int i,number,isum=0; float mean1,count=0.0;

while((scanf("%d",&number)==1))

{

isum+=number; count+=1.0;

}

mean1=isum/count; return(mean1);

}

Exercise (b)

#include <stdio.h>

float add\_3(float value\_1, float value\_2, int value\_3)

{

float result; result=(value\_1+value\_2)\*value\_3;

return(result);

}

void main()

{

float val\_1,val\_2; int val\_3;

printf("Enter three integers "); scanf("%f %f %d",&val\_1,&val\_2,&val\_3);

printf("Result=%f \n",add\_3(val\_1,val\_2,val\_3));

}

Exercise (c)

#include <stdio.h> void ch\_test(void); void main()

{

ch\_test();

}

void ch\_test(void)

{

char ch;

printf("\n Please enter a character: "); ch=getchar();

if (ch>='a' && ch<='z'){

printf("\n a lower case latter was entered\n"); return;

}

if (ch>='A' && ch<='Z'){

printf("\n a upper case letter was entered\n"); return;

}

if (ch>='0' && ch<='9'){

printf("\n a digit was entered\n"); return;

} else

}

printf("\n you entered some other character\n"