Fundamentals of Computing

Week 9

What have we done so far

What we have done so far …

Wk	Lecture
1	Introduction, MDF, Canvas, Assessment Variables
2	C – variables, statements
3,	Input and Output
4	Decision statements
5, 6	While, do-while and for loops
7,8	Arrays

Weeks 9, 10 and 11 …

Functions

Learning Outcomes

1. Learn about functions and their importance.
2. Learn how and when to use functions.
3. Be able to write modular programs using functions.

Functions

Think of baking a cake.

How many people are involved?

What activities are required?

What about if it's a …

Functions

There are two types of functions

➡ library functions (standard functions available on all C implementations)

➡ user defined functions

Library Functions

C language, like any other programming language, is a collection of library functions. They are declared in header files.

Example: `printf` is declared in `stdio.h`

The most commonly used header files are:

<stdio.h>	defines I/O routines
<string.h>	defines string manipulation routines
<math.h>	defines mathematical routines
<stdlib.h>	defines number conversion, storage allocation and similar tasks
<time.h>	defines time-manipulation routines
<conio.h>	defines console input/output routines

Functions

We can use the built-in C functions and/or create our own functions.

We include in functions code that has to be executed on demand.

A function is written once and can be called from anywhere within a page.

Functions

A function is a block of self contained code that performs a specific task. It has an unique name and it can be invoked from other parts of a program.

It optionally returns a value to the calling program.

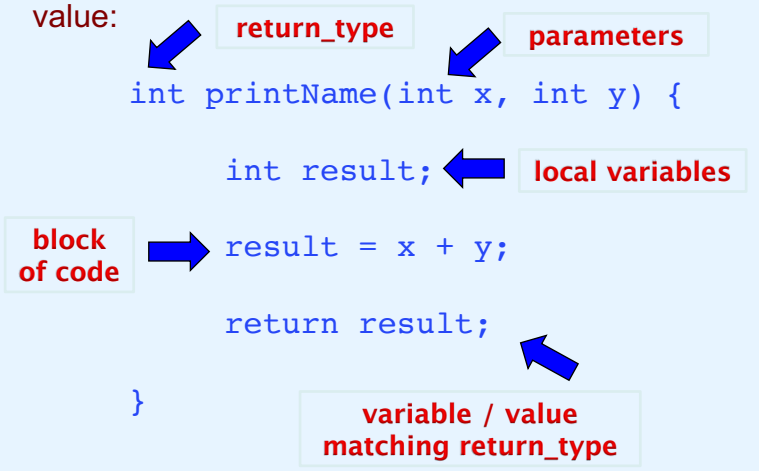
It can optionally accept external data through one or more parameters. Variables passed to it are called arguments.

A function **has to be** defined or declared **before it is used**.

Any C program contains at least one function that has to be called main().

Functions

Example of a function, with parameters, that returns a value:



Functions – definition

A function that returns a value has the following layout:

```
return_type function_name
      (list of comma separated arguments) {

      /* local declarations */
      /* block of statements to execute */

      return return_value;

}
```

Functions – definition

A function that returns no value has the following layout:

```
void function_name(list of arguments) {

     /* local declarations */
     /* block of statements to execute */

}
```

Calling a function

Calling a function in C simply involves referencing its name with the appropriate arguments. The C compiler checks for compatibility between the arguments in the calling sequence and the definition of the function.

Calling a function

Call a function that returns a value:

```
int sum(int a, int b)
{
   int res = a + b;
   return res;
}

int a, b, result;
result = sum(a, b);
```

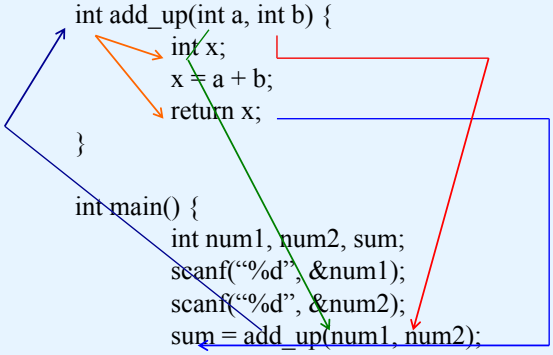
Calling a function

Call a function that returns no value:

```
void sum(int  a, int b)
{
     int res = a + b;
     printf("sum = %d", res);
}

int a, b;
sum(a, b);
```

Functions

```
#include <stdio.h>
#include <conio.h>

int add_up(int a, int b) {
        int x;
        x = a + b;
        return x;
}

int main() {
        int num1, num2, sum;
        scanf("%d", &num1);
        scanf("%d", &num2);
        sum = add_up(num1, num2);
        printf("Sum=%d", sum);
        getch();  return 0;
}
```



1. main() runs
2. num1, num2, sum are declared in memory
3. num1 and num2 values are read from keyboard
4. sum is assigned the value of add_up(num1, num2)
    1. add_up() runs
    2. x is declared
    3. x is assigned a + b
    4. x's value is returned
    5. add_up() ends, main() resumes
5. sum has the x's value
6. sum is printed to screen
7. main() ends

Tasks

1. Using the program Calculator, break it into as many functions as necessary. Each function should contain code that deals with exactly one operation.

2. Write a program that generates a 15 symbol pattern through the use of multiple Functions. Each pattern row is determined by a Char read from the keyboard.

Input: *
Output: ***************
Input: =
Output: =========

Tasks

3. Write a function with one parameter, an integer, that returns 0 if the integer is even and 1 if it is odd.
   Write a program that reads an array of numbers and prints the even numbers by calling the function.

Input: 2 4 1 6 3 19 34
Output: 2 4 6 34