Functions

Dr. Nachiket Tapas

What is a function?

- An independent, self-contained entity of a C program that performs a well-defined task.
- It has
  - Name: for identification
  - Arguments: to pass information from outside world (rest of the program)
  - Body: processes the arguments and do something useful
  - Return value: to communicate back to outside world
    - Sometimes not required

Why use function?

```
void main() {
        int a, b, c, m;
        //code to read a, b, and c

        if(a>b) {
                if(a>c) m = a;
                else m = c;
        }
        else {
                if(b>c) m = b;
                else m = c;
        }
        //print of use m
}
```

```
int max(int a, int b) {
        if(a>b)
                return a;
        else
                return b;
}
void main() {
        int a, b, c, m;
        //code to read a, b, and c

        m = max(a, b);
        m = max(m, c);
        //print or use m
}
```

Lots of related/unrelated task to perform

- Divide and Conquer
  - Create well defined sub tasks
  - Work on each task independently
- Reuse of tasks
  - Phone and SMS apps can share dialler

Advantages

- **Code reuse**: allows us to reuse a piece of code an many times as we want, without having to write it.
- **Procedural Abstraction**: Different pieces of your algorithm can be implemented using different functions.
- **Distribution of Tasks**: A large project can be broken into components and distributed to multiple people.
- **Easier to debug**: If your task is divided into smaller subtasks, it is easier to find errors.
- **Easier to understand**: Code is better organized and hence easier for an outsider to understand it.

We have seen functions before

- main() : special function
- scanf(…), printf(…) : standard input-output functions
- sqrt(…), pow(...) : math functions

Syntax (Function Call and Function Definition)

```
<return type> <function name>(<arguments>) {
        <body>
}

Example:
//This is called function definition
int max(int a, int b) {
        if (a>b)
                return a;
        else
                return b;
}
```

```
void main() {
        int x, c, d;
        c = 6;
        d = 4;
        x = max(c, d); //This is called function call
        printf("%d", x);
}
```

Arguments

- Formal Arguments (function definition)

```
int max(int a, int b) {
        ...
}
```

- Actual Arguments (function call)

```
int main() {
        ...
        x = max(c, d);
        ...
}
```

Arguments

- Input to the function
  - should have matching type
  - type should be declared

- A new copy of these arguments is made (aka Call-By-Value)
  - function works on these new copies

What happens in this case?

```
int max(int a, int b) {
        if(a>b) {
                a = a + 2;
                return a;
        }
        else
                return b;
}
```

8

4

```
void main() {
        ...
        a = 6;
        b = 4;
        x = max(a, b); //x=8
        printf("%d", x);
        printf("%d", a); //a=6
        ...
}
```

6

4

Returning from a function: type

- Return type of a function tells the type of the result of function call.
- Any valid C type
  - int, char, float, double, ...
  - void
- Return type is void if the function is not supposed to return any value

```
void print_value(int val) {
        printf("%d", val);
        //what if we return some value here?
        return;
}
```

Return statement

- If return type is not void, then function should return a value:
  - return return_expression;

- If return type is void, then either no return statement or return statement without any expression.
  - return;

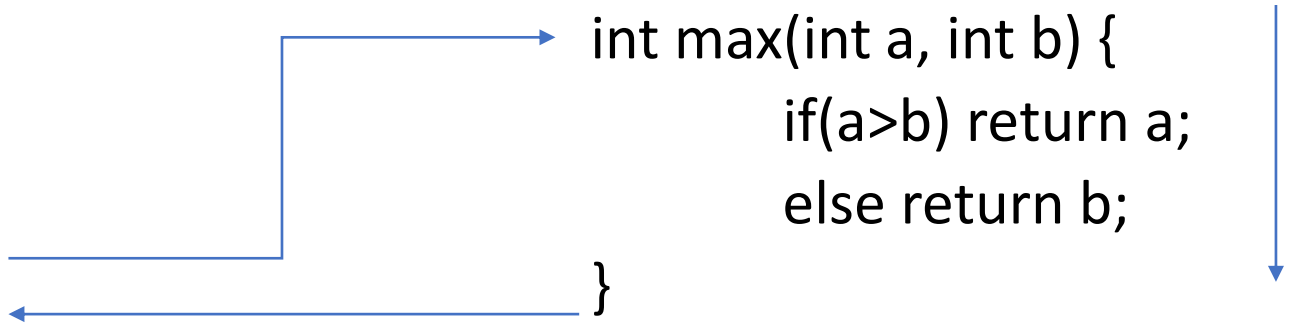
```
void print_positive(int value) {
        if(n<=0) return;

        printf(“%d”, n);
        //something extra
}
```

What happens with return statement?

- When a return statement is encountered in a function definition
  - control is immediately transferred back to the statement making the function call in the parent function

```
int main() {
    ...
    ...
    c = max(x, y);
    ...
    ...
}
```

```
int max(int a, int b) {
        if(a>b) return a;
        else return b;
}
```



How many return statements?

- A function in C can return only ONE value or NONE
  - Only one return type (including void)

  - int main(){

```
        return 0;
}
```