

## More on Functions

### 1 Returning Values

The value is returned by a `return` statement which is usually (but does not have to be) at the end of the function. A `void` function may also have a `return` statement for premature termination, though `returns` not at the end should be used sparingly.

Here is example code for a function which returns the minimum of two values:

```
int minimum ( int v1, int v2 ) {
    if( v1<v2 )
        return v1;
    else
        return v2;
}
```

Note the `v1` and `v2` exist only inside the function: their *scope* is *local*. If one were to change the value of `v1` in the function, it would NOT affect the value of the variable that was passed to it.

### 2 Prototypes

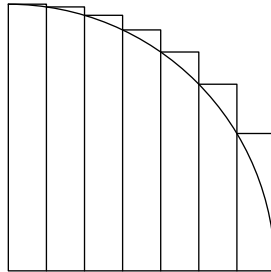
*Prototypes* are used to tell the compiler about the functions that are available. A prototype might be

```
int minimum(int,int);
```

(Note the semi-colon!) Actually including the names of the parameters helps the reader. Prototypes are needed if the call to the function occurs before the function code. Some advocate prototypes for all functions. Ideally these should be in a *header* file, discussed later.

### 3 Sample Program: archimedes.c

Archimedes tried to estimate  $\pi$ . His idea was to consider a quarter of the circle (which has area  $\pi/4$ ). Then he divided the circle into parallel vertical strips and estimated the area as the sum of the strips. If one takes the left-endpoint (as below) one gets an over-estimate. The program also goes awry when the divisions are small, due to numeric problems.



```
// program to replicate Archimedes' calculation of pi
// also shows numerical problems - wdg 2008
#include <stdio.h>
#include <math.h>

float estimate(int); // prototype

int main(void) {

    int divs;
    for(divs=1; divs<2000000; divs*=2) {
        float pi = estimate(divs);
        printf("For division=%d Pi is approximately %.5f\n", divs, pi );
    }
    return 0;
}

float estimate(int divisions) {
    int i;
    float sum =0.0, width = 1.0/divisions;
    for( i=0; i<divisions; i++ ) {
        sum += width * sqrt( 1.0 - (i*width)*(i*width) );
    }
    return 4*sum;
}
```