DATA TYPES AND OPERATORS

BEESHANGA ABEWARDANA JAYAWICKRAMA

UTS:
ENGINEERING AND
INFORMATION
TECHNOLOGY

feit.uts.edu.au

PRIMITIVE DATA TYPES IN C

Most commonly used primitive data types:

Data type	sizeof()	Min	Max
char	1 byte	-128	127
int	2/4 bytes	-32 768 -2 147 483 648	32 767 2 147 483 647
float	4 bytes	1.2e-38 (±)	3.4e+38 (±)
double	8 bytes	2.3e-308 (±)	1.7e+308 (±)

char – stores exact numbers in 2's complement notation. ASCII table maps binary representation to a character

int – stores exact numbers in 2's complement notation.

float/double – stores approximate numbers in IEEE floating point notation (sign, exponent, mantissa)

DATA IS STORED IN MEMORY AT ADDRESSES

c	i				d				
C	0				3.14159265359				
0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19

			f						
			2.12						
0x1A	0x1B	0x1C	0x1D	0x1E	0x1F	0x20	0x21	0x22	0x19

```
char c = 'C';
int i = 0;
double d = 3.14159265359;
float f = 2.12;
```

OPERATORS

+	addition
−	subtraction (addition of 2's complement)
*	multiplication
/	division
%	remainder/modulus

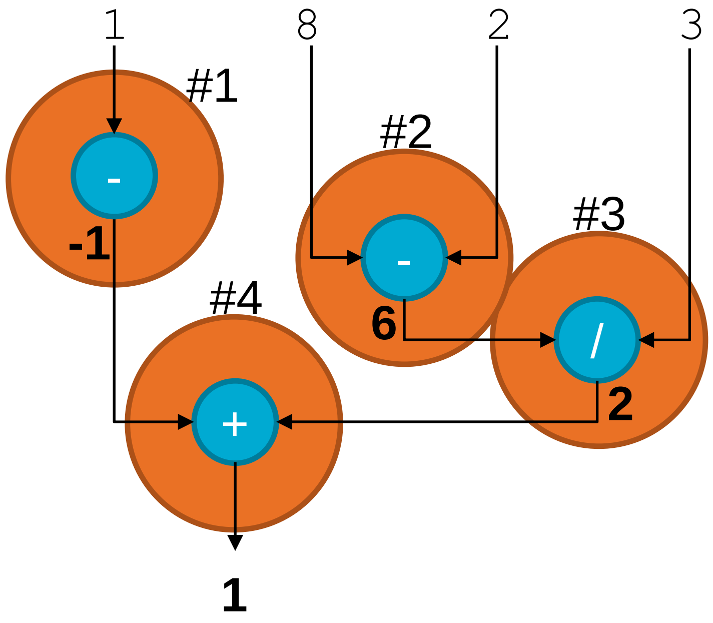
Order of precedence: execute left to right following the order below

High	Paranthesis		
↓	+	−	(unary)
	*	/	%
	+	−	(binary)
Low	=		

ORDER OF PRECEDENCE EXAMPLE

```
int a=1; int b=8; int c=2; int d=3;

-a + ((b - c) / d)
```



Result: 1

Process only two values at a time.

The tree structure is called the evaluation tree.

DIVISION AND TYPE CAST

```
int a = 3; int b = 2; double c;
c = a/b;
```

Since a and b are integers, the operation is integer division. c = 3/2 = 1.

However, 3.0/2 = 3/2.0 = 3.0/2.0 = 1.5

If integer division is not the preferred operation use type cast

`c = (double) a / b;` -OR-

`c = a / (double) b;` -OR-

`c = (double) a / (double) b;`

Type casting does not change the original data type of a and b variables.

UNIVERSITY OF TECHNOLOGY SYDNEY

FUNCTIONS

BEESHANGA ABEWARDANA JAYAWICKRAMA

UTS:
ENGINEERING AND
INFORMATION
TECHNOLOGY

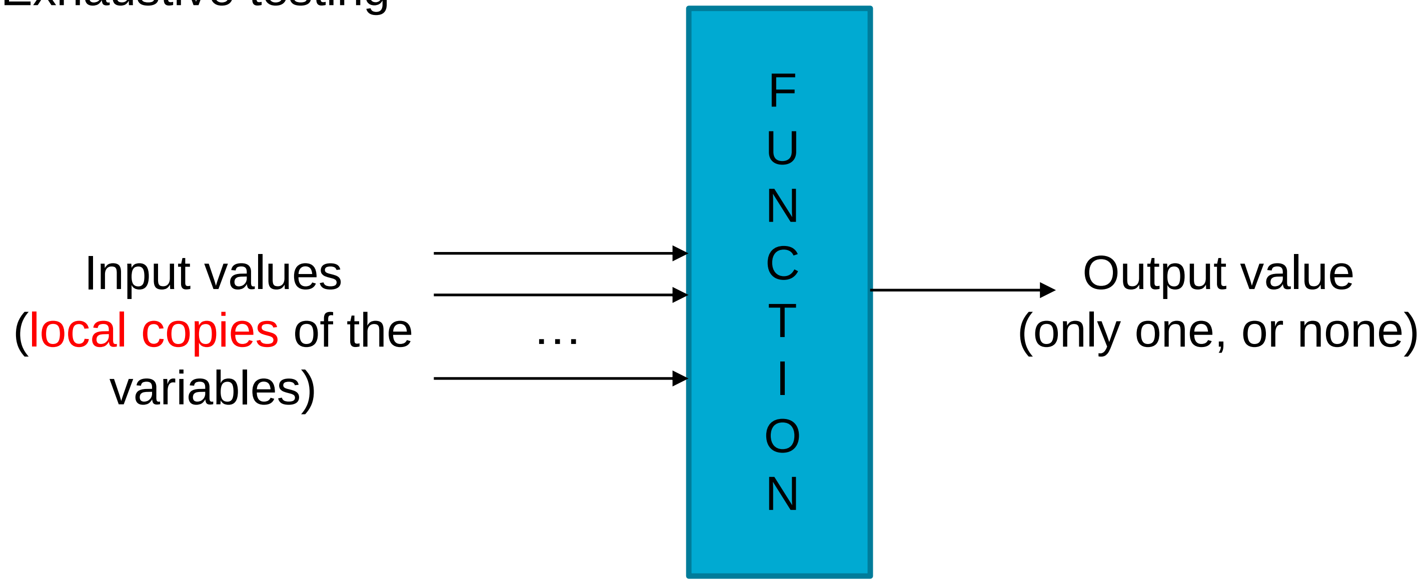
PROCEDURAL PROGRAMMING

C is a procedure (function) oriented programming language. Every C program has at least one function – main().

Functions allow us to logically group statements that performs a specific task.

Breaking down a program to sensible number of functions is important.

> Human readability of the code
> Code reuse
> Exhaustive testing



FUNCTION PROTOTYPES

When using variables there were two stages – definition/declaration, initialisation. Functions also have two stages

> Prototype – tells the compiler the function name, data types of input parameters (if any) and the data type of output parameter (if any)

> Definition – implementation of the function

Two types of functions

> Predefined – someone else has written the function prototype and definition, and you use it

> User-defined – you write the function prototype and definition, and you use it. Possibly will become a predefined function to someone else. Almost all non-trivial C programs have user-defined functions.

USER-DEFINED FUNCTIONS

```
#include <stdio.h>
#define KMS_PER_MILE 1.609

double mi_to_km (double mi);
```

Function prototype

```
int main(void)
{
  double miles;
  printf("Enter miles> ");
  scanf("%lf", &miles);
  printf("%lf miles = %lf kms", miles, mi_to_km(miles));
  return 0;
}

/*********************************************************
Convert miles to km
inputs:
  mi - the number of miles
output:
  equivalent number of kms
*********************************************************/
```

Block comment explaining
- What function does
- Inputs
- Output

```
double mi_to_km (double mi)
{
  return KMS_PER_MILE*mi;
}
```

Function definition (implementation)

FAQ

How do we decide what goes in main, what goes in user defined functions?

> Group together all statements in a program that are intended to do a specific task and put them into functions – don't overdo it either, be sensible.

> A common sign is that if you have the need to copy paste some lines of your code, you must have put that into a function. But keep in mind this is not the only sign.

How can I go about writing a user-defined function?

> Understand what the function is expected to do

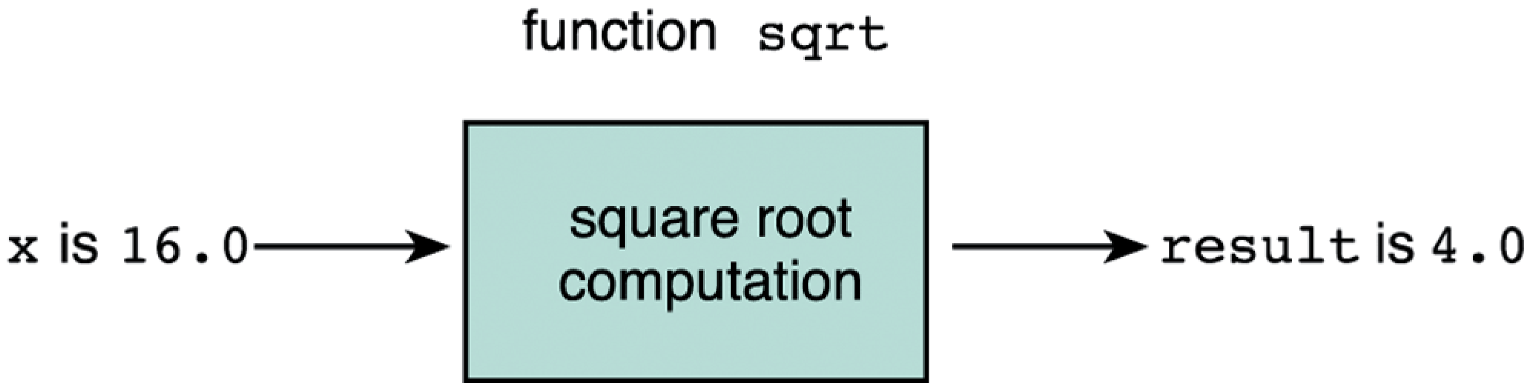
> Determine the inputs and the output, and write the function prototype

> Think about how to implement, and write the function definition

> TEST it thoroughly – most students forget!

EXAMPLE

Who knows how to calculate the square root of a number?



Used for example as …

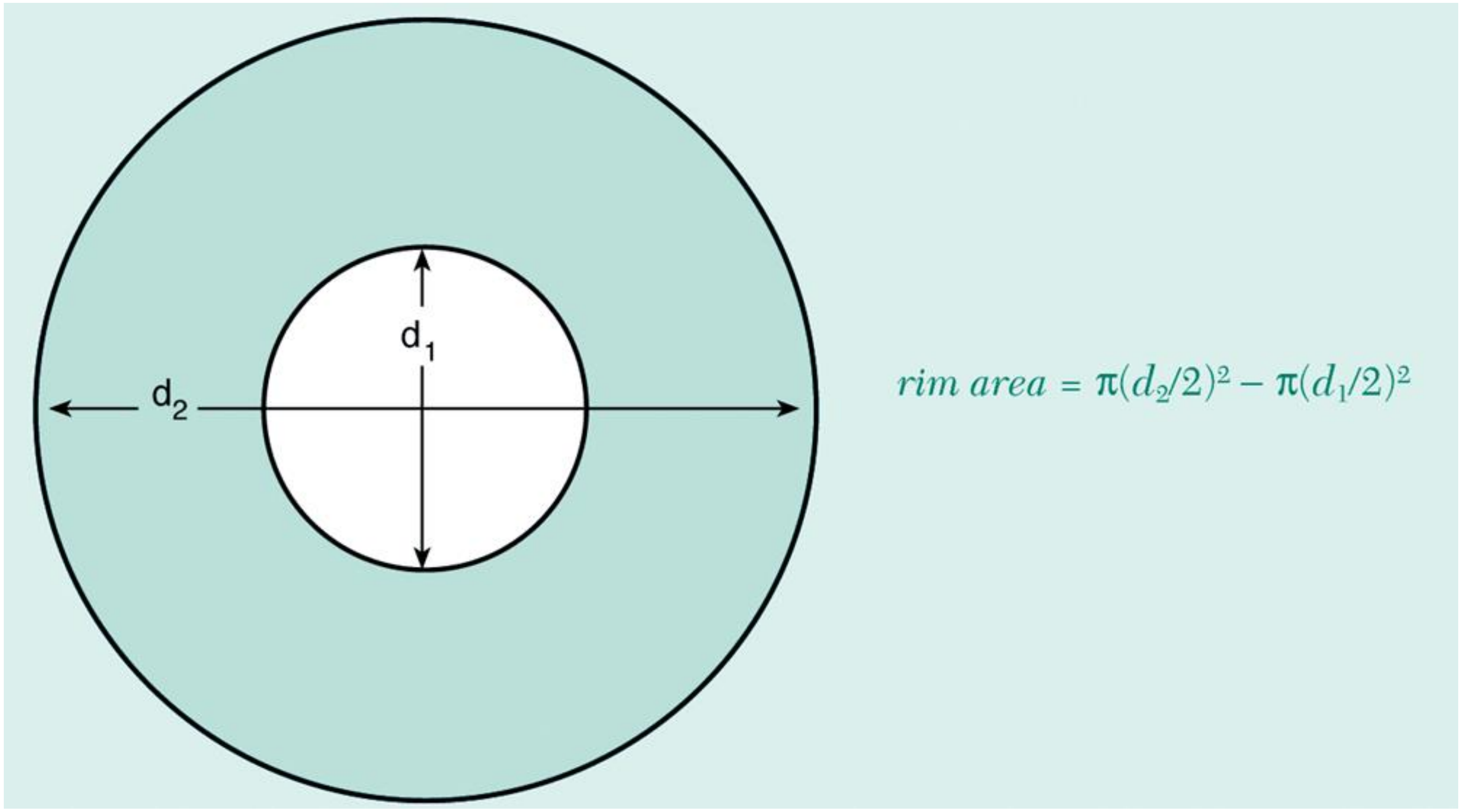
```
z = 5.7 + sqrt(16.0);
```

… this will be evaluated as …

```
z = 5.7 + 4.0;
```

EXAMPLE

Calculate the area of a flat washer



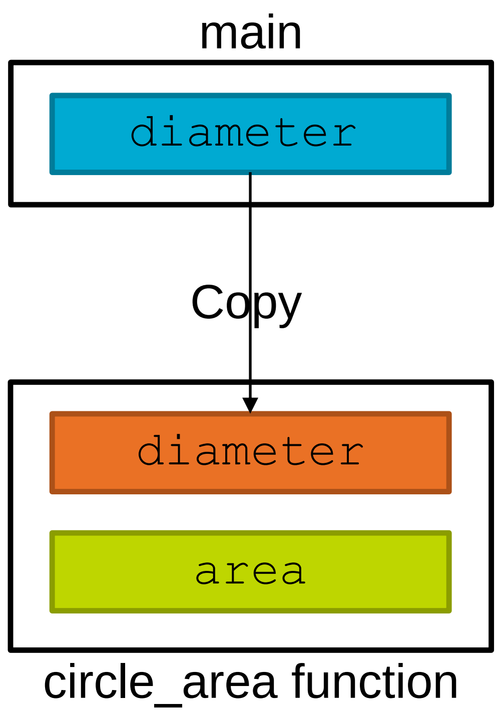
$$rim\ area = \pi(d_2/2)^2 - \pi(d_1/2)^2$$

FUNCTIONS MAKE A LOCAL COPY OF INPUT PARAMETERS

Any change made to the diameter inside the function will not change diameter inside the main.

This is a key concept originating from how C handles scope of variables – more to come.

The local copies of the variables inside the function are created in the beginning of the function execution, and destroyed when the function returns.


main
diameter
Copy
diameter
area
circle_area function