**Module2**

Pointers

A pointer is a variable that contains a memory address.

Example:

int a;

int b = 10;

float c = 5.8;

printf (“address of a is at %u \n”, &a); *note: %u = unsigned integer (always shows positive interpretation, -ve🡪+ve)*

printf (“address of b is at %u \n”, &b);

printf (“address of c is at %u \n”, &c);

int a has no data, is at memory location xxxxxx1 (takes 4 bytes)

int b has value of 10, is at memory location xxxxxx2 (takes 4 bytes)

float c has value of 5.8, is at memory location xxxxxx3

Their values are actually stored as binary, in zeros/ones.

-----------------

int p1; *a variable*

vs.

*(declaration)*

**int \*p1;**  *becomes a pointer variable* *(contains a memory address)*

*(initialising)*

**p1 = &a;**  *“pointer p1 contains the address of a” the address of a = the memory location stated above*

*(declaration + initialising)*

**int \*p2 = &b;** *“pointer p2 contains the address of b”* *same as the above statement*

**float \*p3 = &c;** *“float pointer p3 contains the address of c”*

-----------------

Putting the \* asterisk before p1 means that I am trying to access the value/data from the memory location, which the address is stored in pointer variable p1.

**Therefore, if you want to access the value of a variable, simply put an \* asterisk sign before the name of the pointer.**

**printf (“the value of p1 is %u”, p1);** *prints the value of pointer p1, which is the memory address of a*

**printf (“the value of a is %d”, \*p1);**  *prints the value of a, the data value of variable a*

**printf (“the value of p3 is %u”, p3);** *prints the value of pointer p3, which is the memory address of c*

**printf (“the value of c is %f”, \*p3);**  *prints the value of c, the data value of variable c  
note the use of %f because variable c is of type float*

-----------------

Sum using pointers

printf (“the sum of a and b is %d”, a + b); *summing a and b WITHOUT pointers*

printf (“the sume of a and b is %d, \*p1 + \*p2 ); *summing a and b WITH pointers*

-----------------

Changing values using pointers

p1 = p2; *assign p1 the memory address value of p2/ p1 stores address of b*

Printf (“value of p1 is %u \n”, p1); *value of p1 = memory address p2 (after assigning p1 = p2)*

Printf (“value of b is %d \n, \*p1); *value of b = 10, using \*p1 (as p1 now has the same address as p2, storing value of b)*

-----------------

Extra notes on pointers

Pointers are variables, so space is allocated for them + they have their own memory addresses.