Day 9 Understanding Pointers

pointer- variable that stores the address of another variable

1. declare a pointer: typename *ptrname; typename is type of variable pointing to

ex: float *value, percent; pointer nested with variable

2. once declared must point pointer at variable: pointer = &variable

& means address of

2 ways to refer to variable that has pointer- 1. rate 2. *p_rate

accessing by using pointer is indirect access or **indirection**

*ptr and var refer to contents of var

ptr and &var refer to the address of var

each byte of memory has its own address so a multibyte variable occupies several addresses

address of variable actually address of lowest byte- variable type tells compiler how many bytes it will occupy

pointers and arrays

array subscripts are really just pointers

array name without brackets is really just pointer to address of first element

array == &data[0]

CAN declare pointer and initialize to point at array:

int array[100], *p_array; p_array = array;

array elements stored at addresses incremented according to variable type size- access successive elements of array by by sizeof(datatype)

incrementing pointers

when increment pointer by one, automatically increases the pointers value so that it points to the next array element

ptr++ to increment

ptr +=4 points to 4 array elements ahead

decrementing pointer is special case where increment by adding negative number

point to first element in array then increment across

cannot perform incrementing and decrementing operations on pointer constants (array name w/o brackets is pointer constant)

if not careful can increment or decrement beyond array- very dangerous can overwrite other parts of memory (OS? program?)

pointer ops

1. **assignment**- assign value to a pointer
2. **indirection**- use * operator to get value stored at location
3. **address of**- use & to get address of a pointer. could have pointers to pointers
4. **incrementing**- add integer to pointer in order to point to different memory location
5. **decrementing**- subtract an integer from a pointer to point at different memory location
6. **comparison**- valid with only 2 pointers pointing at same array. ptr1 < ptr2 is true if pointing to lower memory location- can use as a condition

must initialize otherwise who the hell knows where it is pointing

*ptr = 12 value 12 assigned to wherever ptr pointed at

*array is array's first element. *(array + 1) is second element

passing arrays to functions

only way can pass an array to a function is by using a pointer

if pass that value to a function function knows the address of the array and can access the array elements using pointer notation

can identify last element of array by storing a special value there. OR pass the function the array size as an argument