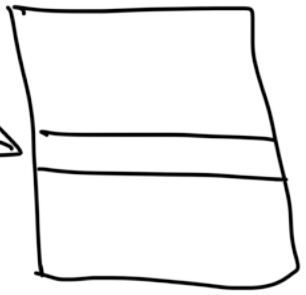
C - Week - 7 Notes

L1 Introduc^{n} to Arrays - I

- Pointers & memory

  &i = address

  *(&i) == data == i



- Lists - use eg. lists to work on homogeneous data.
- Array - convenient short cut, specify start of list, no. of entries, how to reach next value.
- int A[10]; ⇒ consecutive memory locations, each entry same size (int)
- int A[10]; → array of 10 int

  const int N = 1000; } array of 1000
  int B[N]; } integers

  short C[4] = {0};

  ↳ 4 16 bit values - init to 0.

  float D[20] = {1, 2, [5] = 3.14, 7}

  1.0, 2.0, 0, 0, 0, 3.14, 7 .....

  0s ⟶

L2 Intro. to Arrays Part - 2

- unsigned long → %lu
- const int N = 1000;
  int A[N];
  sizeof(int) = 4
  sizeof(A) = 4 x 1000 = 4000
  No. of elem. in A = sizeof(A) / sizeof(int)
  = 1000
- int A[4] = {1, 2, 3, 4}

  0 → 1
  1 → 2
  2 → 3
  3 → 4

  int B[4] = {[2] = 7, 8}

  0 → 0
  1 → 0
  2 → 7
  3 → 8

L3 Pointer Arithematic

- Address → loca^n in memory acts as an index. Offset from some starting pt.
- Base → add. of 1st loca^n in list.
- Offset → Increment by # bytes to get to nth loca^n.
- A[i] = *(A+i)

  i → offset in terms of no. of elements

int *p → p++ → increment by 4 bytes
char*p → p++ → increment by 1 byte

- Array bounds → int A[10];
  if you try to access A[11] ? It translates into memory access. C does not enforce bounds.
- It is not a good practice. Leads to segmentation fault, segmentation violation, prog. crash, inappropriate access (security violation).
- int A[10]; int B[10];
  In most cases, memory will be allocated continuously; but there is no guarantees.

L4 Eqs. of Pointer Arithmetic

eg. to show the no guarantee if a memory is allocated continuously or not.

L5 Strings in C

- no string datatype, it is simply an array of characters. Char: usually ASCII (8-bit). Terminated by NULL (0 byte)

- char [100] s1 ; → 100 bytes
  char * s2 = "Hello" ; → 8 bytes
  char s3[] = "_____" ; → 12 bytes
- pointers & sizes have to be taken care.
- security & safety → bounds checking not enforced. Read past end of string is possible. Malicious inputs from users, files. Many such issues.

L6 Starting Functions

- strlen → compute len of string
- strnlen → more safer, will not spill out of memory
- strcpy (dummy, arr) → copies arr to dummy.
- strcat (a, b) → concatenates 2 arrays
- strcmp → compare strings for equality
- Unicode → wide characters, multi-byte encoding, needed to handle range beyond ASCII. Replacement functions like wcslen (in place of strlen). Pretty OS & lib. dependant. Not part of language specification.