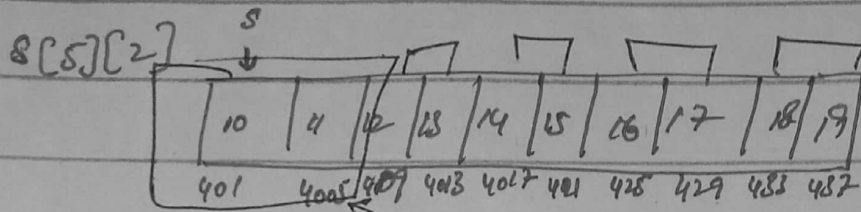


# Pointer in C Language.



$$s[0] = *(s+0) = 401$$

$$s[1] = *(s+1) = 409$$

⇒ By using  $s[i]$  we only accessed base addresses i.e., 401, 409, ... 437.

$$s[4] = *(s+4) = 421$$

In two dimensional Array

$$s[2][1] = *(s[2] + 1) = *(*(s+2) + 1)$$

int - 4 bytes

char 1 byte

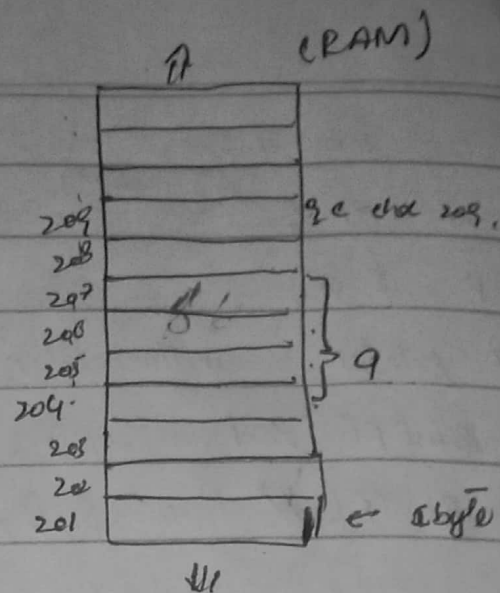
float 4 bytes

```
int a;
```

```
a = 5;
```

```
...
```

```
a++;
```



Pointer - variable that stores address of another variable

```
int a;
```

```
int *p;
```

```
p = &a;
```

```
printf("p = %d", p);
```

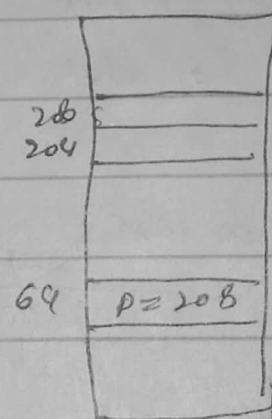
```
printf("%d", *p);
```

```
printf("%d", p);
```

```
printf("%d", *p); // dereference
```

```
*p = 8; // change
```

```
printf("%d", a);
```



✓



9



1

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

10



If this would pointer to an  
character then computer only

See at 1 byte that starts

200 or

Generic Pointer

void \* P; ;

P = P ;

int

int

P =

//

Point

Per

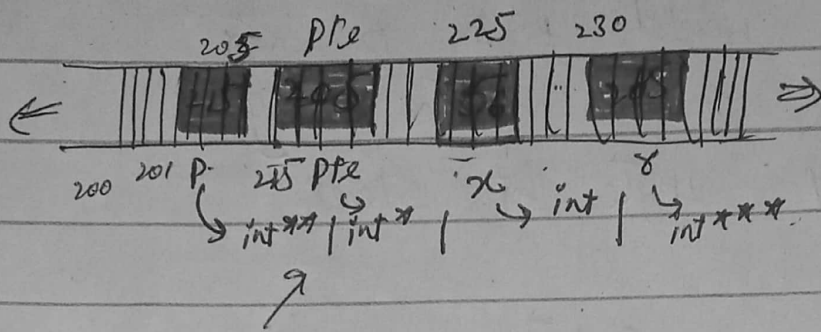
NR

log

cha

in

# Pointer to Pointer



```
int x = 5;
```

```
int* ptr;
```

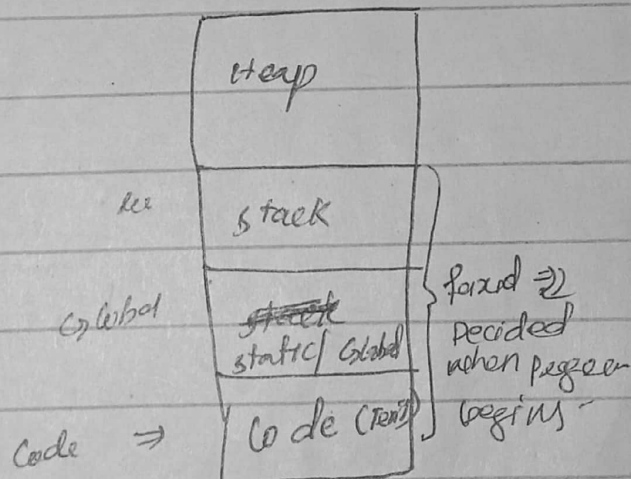
```
*ptr = 6;
```

```
int **pp;
```

```
pp = &ptr;
```

```
int ***p;
```

```
p = &pp;
```



# Pointer and Arrays

$*A[i] = (A+i)$

$A[i] = *(A+i)$

When array in a function

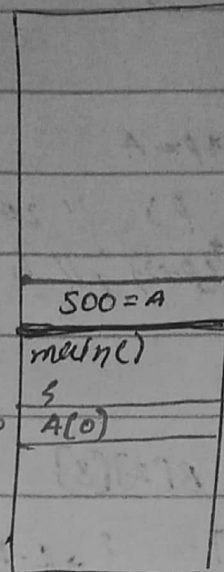
e.g;

```
int SumOfElements (int A[])
```

```
{
```

↙  
this means in 'A'  
backend-  
int\* A

Stack



```
{
```

```
int main()
```

```
{ A[0] = { 1, 3, 2, 4, 5 } ;
```

```
SumOfElements (A) ;
```

→ goes address

## Character Arrays and Pointer.

```
void Print (char *C)
```

```
{ while (*C != '\0')
```

```
{
```

```
printf("%c", *C);
```

```
C++;
```

```
}
```

```
}
```

Print

Z

```
char *P = "Hellow"; // Valid statement
```



# pointers And multidirectional Array

int A[5];

200	204	208	212	216
2	4	6	8	10
A[0]	A[1]	A[2]	A[3]	A[4]

int \*p = A  
 printf p; // 200  
 printf \*p; // 2  
 printf \*p+2; // 6  
 ...  
 \*(A+i) same A[i]

int B[2][3]

400	404	408	412	416	420	424
2	3	6	4	5	8	
B[0][0]	B[0][1]	B[0][2]	B[1][0]	B[1][1]	B[1][2]	

B[0] } → 1-D array  
 B[1] } of 3 integers  
 2D-array

~~int \*p = B; // This will point to 1D Array~~  
 // It is 8 integers

int \*p[3] = B; // valid  
 This is pointer array that  
 point 3 integers pointers

Print B or &B[0] // 400  
 Print \*B or B[0] or &B[0][0] // 400  
 Print B+1; // 412  
 Print &B[1]; // 412  
 Print \*(B+1); // 412  
 Print B[1]; // 412  
 Print &B[1][0]; // 412

$$B \cong *B[0]$$

$$*B \cong B[0] \text{ or } *B[0][0]$$

$$(B+1) \cong *B[1]$$

$$*(B+1) \text{ or } B[1] \text{ or } *B[1][0]$$

↳ returning  $\text{int}^*$  to first integer in  $(B[1])$

$$\text{Print } *(B+1) + 2 ; // 420$$

↳  $\text{int}^*$

$$\rightarrow B[1] + 2 \text{ or } *B[1][2] // 420$$

$$*( *B + 1 ) ;$$

$$= *(B[0] + 1) ;$$

$$= *( *B[0][1] ) ; = B[0][1] = 3$$

Note

$$B[i][j] = *(B[i] + j)$$

$$= *( *(B+i) + j )$$



## DYNAMIC MEMORY ALLOCATION

heap = free store of memory (large free pool of memory)  
↳ Dynamically memory

\* C Functions

\*\*\* malloc

void\* malloc(<sup>assigned int</sup>size\_t size)<sup>\*n</sup>

\*\*\* calloc

void\* calloc (nofblock, sizeof(int));

\*\*\* realloc

void\* realloc (void\* ptr, nofblock);

\*\*\* free (ptr);

↳ If free all block of memory and their remain garbage values.

## Pointers as Function returns

Let a function

```
int add(int a, int b)
{
    return a+b;
}
```

then pointer to this function  
is

```
int (*P)(int, int)
P = &add;
```

Then we can call

```
printf("%d", *P(5, 6));
```

## Function Pointers and callbacks

Function pointers

↳ can be passed as argument  
to functions.