sizeof(char) = 1

→

```
struct A {
        int x;
        char y;
      };
```

no space.
↓
a . x = 5; ✓

struct A a;
→ static memory allocation.

a → x = 5; ← X

(1) struct A $\underbrace{*p}$; $\xrightarrow{?}$ (3) p → x = 5; ✓

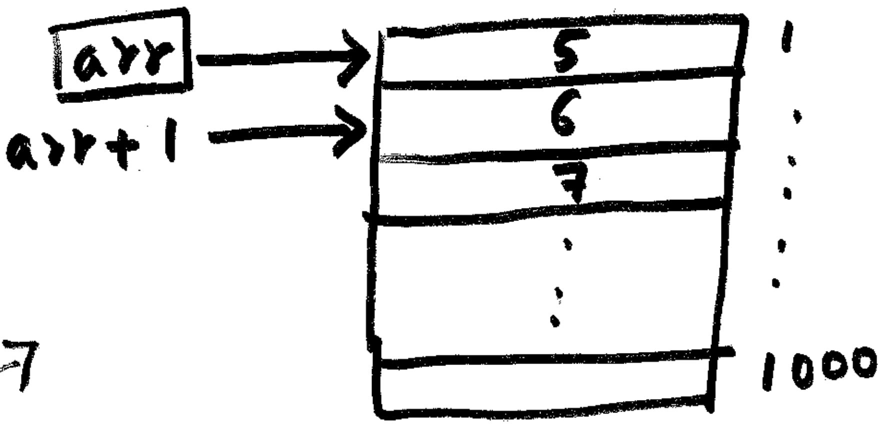
(2) p = (   ) malloc (sizeof (struct A)) ;

↙
dynamic memory allocation.

→ int arr[1000]; /* arr - space allocated to 'arr' equals size of 1000 integers */

int x = *(arr+1);

/* x = 6 */



→ int *arr; /* only 4 bytes allocated at compile time */

```
arr = (int *)malloc(sizeof(int) * 1000);
```

return type = void *.

→ /* use arr */

free(arr);

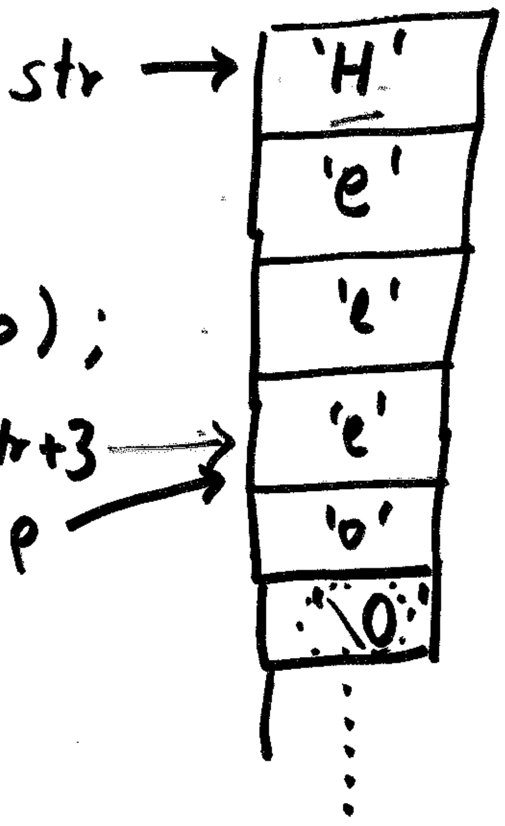
if (arr) free(arr);

⇕

if (arr != NULL)

```
char letter;
letter = 'c';
char str[10];

strcpy(str, "Hello");
```



```
'A'   char *p = str+3;
      printf("%c\n", *p);
           (l)

int a = 6, b = 5;
 if (a == b)  printf("Equal");
 int c = 6;
 if ((a != b) && (a == c)) printf("Equal")
```

↑
logical AND.   logical OR ||

```
int d = (a == b);          /* 0 */.

if (a == b)   1; else 2;
```

Bitwise operators.

```
int a = 3;
int b = 2;
int c = a & b;   /* 2 */
c = c << 2;      /* 8 */
c = c >> 1;      /* 4 */
```

```
a = ...00011
b = ...00010
c = ...010

......0010
00... 1000
```

```
int d = 2;
    d += 4;    /* 6 */
    d++;       /* 7 */
   --d;        /* 6 */
    d--;       /* 5 */

int e = d--;
int c =  (d == 5) ? 7 : 8;
```

```
const int a = 3;
```

scope:

```
        int a = 3;
block. → {
            int a = 4;   → scope is limited to inside the block.
  4 ←       printf("A = %d", a);
         }
  3 ←    printf("A = %d", a);
```

switch:

```
     int a = 2;
   switch (a) {
      case 1:  printf(" 1 "); break;
      case 2 : printf("2"); break;
      default: printf ("default");
               break;
   }.
```

```
→ int a = 5;
→    while (a < 7)  {
                          printf(" Hello \n" );
                              a++;
  /* 2 times */.           }.

→ int *p = &a;
     printf("a = %d", *p);  /* 7 */
```

c preprocessor.

```
#include <stdio.h>

main()
 {
 }

#define    WIDTH. 100      ←.

                                #if.

           A.
```

command_line arguments.

```
int main(int argc, char *argv[])
```

$\nearrow$

Find Error.

```
int  A[12];                      /* A[0] ... A[11]
                                                 */
int  i;
for (i = 0; i <= 12; i++) {
        A[i] = i+1;


/* A[12] = 12+1; */
```

Lab login info. R. Random

username: rxrandom ←.

DOB : 01/12/1990

password: ~~rxr 01 12 1990~~.
rxr 1990 01 12