```c
struct s
{
    int data;
    char c;
}

struct s *ps;

ps = &s-1;

ps->data = 10;
ps->c = 'K';

s-1->data = 5;
```
`chan a [3][10] = \{ "Hello" \};`

`char *p[3] = \{ "hello", "i", "j" \};`

```
int a[10];
int *a;
int a[10][20];
int *a[];
```
```c
int a[10];
```

1). size fixed

2). insert at beg/middle

3). int a[1000]; 

\[ 1000 \times 4 = 4000 \text{ bytes} \]

\[ 40 \times 4 = 160 \text{ bytes} \]

Dynamic Memory Allocation.
`void* malloc(size of bytes)`

`int* a = malloc(n * size of (int))`
The diagram illustrates the memory organization of a computer's memory stack and heap. The top section is labeled "code/text" and contains instructions. Below that is the "data" section, which includes "heap" and "global variables." Further down is the "stack," which includes local variables, parameters, a return address, and a function pointer.
Stack

a

$\texttt{a} = \texttt{NULL}$

Heap

$\texttt{OX100}$

$\texttt{main()}$

$\{ $

$\texttt{int* a; }$  

$\texttt{a = malloc(...);}$  

$\texttt{func(a);}  \}$

$\texttt{func(int *p)}$

$\texttt{p[0] = 1; }$

$\texttt{3}$

$\texttt{free(a)}$
struct node {
    int data;
    struct node *next;
}

3