Structs can be created dynamically in the heap and pointers can be assigned to point to them.

For example:

```c
// Definition of struct electric
struct electric
{
    string current;
    int volts;
};
```

The following two pointers can point to struct variables of type `electric`:

```c
electric *p, *q;
```

Let’s create two structs (nodes) in the heap and make these pointers point to them:

```c
p = new electric;  q = new electric;
```

Now, there is garbage in these two nodes. To put data inside them, we must be able to access their `current` and `volts` members. As usual, we use the (*) to access the whole node using its pointer, and the dot (.) to access a particular member of the struct:

```c
*p.current = “DC”;  *p.volts = 110;  *q.current = “AC”;  *q.volts = 220;
```

We can also write the above statements as follows:

```c
p -> current = “DC”;  p -> volts = 110;  q -> current = “AC”;  q -> volts = 220;
```

Here, the (*pointer.) is replaced by (pointer ->)

---

**Question:** Can a pointer be a member of a struct?

**Answer:** Yes. Here is an example:

```c
struct node
{
    int info;
    node *next;
};
```

```c
node *p;
```

```c
// Declare a pointer to a node
```

```c
p = new node;
```

```c
// Create new node and let the pointer point to it
```
Nodes containing pointers can be used to build dynamic **Linked Lists**:  

![Diagram of a linked list](image)

**Example:**
Assume the following declarations:

```c
struct node { char info; node *next};   node *p;   node *h = NULL;   node *c = h;
```

Draw a diagram representing the nodes and pointers generated by the following code:

```c
string s = "DATA LIST";
for (int i = 0; i < s.length(); i++)
{
    p = new node;  p->info = s.at(i);  p->next = NULL;
    if (c == NULL) { h = p;  c = p;} else { c->next = p;  c = c->next; } 
}
```

**Solution**
The given code will insert the characters of the string `s` into a linked list of nodes in the same order, i.e.,

![Diagram of linked list with characters](image)
Assume the following declarations:

```c
struct node { int info; node *next}; node *p;
```

Draw a diagram representing the nodes and pointers generated by the following code:

**Solution**

The above code will insert the digits of the integer \( n \) into a linked list of nodes, with each digit in a separate node. The first node will contain the most significant digit and the last node will contain the least significant digit.