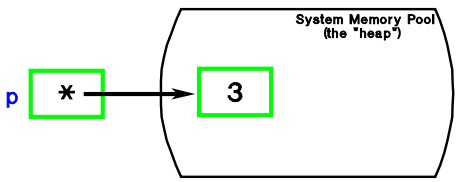
C++ Dynamic Memory

Memory can be allocated dynamically (i.e., during program execution). The requested memory does NOT have a name (like a regular variable), but instead it is referenced via a pointer.

```
int *p;

p = new int;
*p = 3;
```



The new operator returns a pointer to the dynamic memory

C++ Dynamic Arrays

```
int *p;

p = new int[100];
for(int i=0; i<100; i++)
   p[i] = 5;
```

NOTE: p is really a pointer (to int), not an array. However, the "subscript" form of pointer arithmetic can be used, as seen in previous examples.

Therefore, it appears that "arrays" can be dynamically allocated!

Freeing Dynamic Memory

After a program is done with dynamic memory, it should be "given back" to the system:

```
delete p;
```

This releases the memory pointed to by p back to the system.

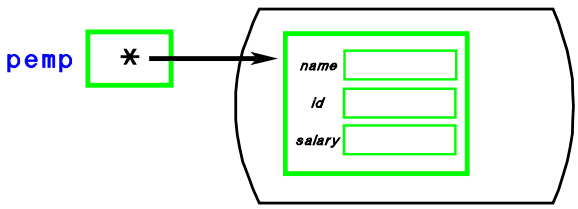
C++ Dynamic structs

```
struct emptype
  {
   char name[20];
   int id;
   float salary;
  };

emptype *pemp;

pemp = new emptype;
```

"pointer to"



C++ Dynamic structs

To access struct members:

```
(*pemp).id = 12345;
strcpy((*pemp).name, "Joe Blow");
(*pemp).salary = 1.98;
```

A new notation:

```
pemp->id = 12345;
strcpy(pemp->name, "Joe Blow");
pemp->salary = 1.98;
```

The operator heirarchy of C makes the ( )'s necessary in the first cases above. The alternate -> notation can be used instead. The two notations are equivalent.

PTR0220

Operator Hierarchy Chart (Complete)

Operators	*Notes*	*Associativity*
`() [] -> .`		Left to Right
`! ~ ++ -- + - * & (type) sizeof`	Unary ops	Right to Left
`* / %`	Arith ops	Left to Right
`+ -`	Arith ops	Left to Right
`<< >>`	Binary shifts	Left to Right
`< <= > >=`		Left to Right
`== !=`		Left to Right
`&`	Logical AND	Left to Right
`^`	Logical EX-OR	Left to Right
`|`	Logical ORt	Left to Right
`&&`		Left to Right
`||`		Left to Right
`?:`	Decision OP	Right to Left
`= += -= /= %= &= ^= |= <<= >>=`		Right to Left
`,`	Comma	Left to Right

DYNAM070