Announcements


int *p, *q;
p = new int;
q = p;
*q = 8;
cout << *p;  // What is output?
q = new int;
*q = 9;
p = NULL;   // Do you like this?
delete q;
q = NULL;    // Do you like this?

Memory leak:

Deleting a null pointer:

Dereferencing a null pointer:
Fun and games with pointers:

```c
int * p, * q;
p = new int;
q = p;
delete p;
... // some random stuff
cout << *q;  
```

Do you like this?___________
Fun and games with pointers:

```
int * p; int x;
p = x;
Do you like this? ___________
What kind of error?
Compiler  Runtime
```

```
int * p; int x;
Variable p can be given a target (pointee) in two ways. Write an example of each.
```

```
int * p; int x;
Use the letters S and H in a meaningful way to tell where the pointee exists in memory.
```

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

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

```
int * p; int x;
Variable p can be given a target (pointee) in two ways. Write an example of each.
```

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

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

```
int * p;
*p = 37;
p = NULL;
*p = 73;
Do you like this? ___________
What kind of error?
Compiler  Runtime
```

```
int * p;
p = x;
Do you like this? ___________
What kind of error?
Compiler  Runtime
```

```
int * p, * q;
p = new int;
q = p;
delete p;
... // some random stuff
cout << *q;
Do you like this? ___________
```
Which of the following snippets are buggy?

```c
int *p, *q;
p = new int;
q = p;
*q = 8;
q = new int;
*q = 9;
p = NULL;
```

```c
int *p, *q;
p = new int;
q = p;
*q = 8;
delete q;
*p = 12;
p = NULL;
```

```c
int *p;
int x = 5;
p = &x;
delete x;
p = NULL;
```

```c
int *p;
int x = 5;
*p = x;
```
Stack vs. Heap memory:

System allocates space for `s` and takes care of freeing it when `s` goes out of scope.

Data can be accessed directly, rather than via a pointer.

```
void fun() {  
    string s = "hello!";  
    cout << s << endl;  
}  
int main() {  
    fun();  
    return 0;  
}
```

Allocated memory must be deleted programmatically.

Data must be accessed by a pointer.

```
void fun() {  
    string * s = new string;  
    *s = "hello?";  
    cout << *s << endl;  
    delete s;  
}  
int main() {  
    fun();  
    return 0;  
}
```
Pointers and objects:

class face {
public:
    void setName(string n);
    string getName();
    ...
private:
    string name;
    PNG pic;
    boolean done;
};

face a, b;
... // init b
a = b;
a.setName("ann");
b.getName();

face * c, * d;
... // init *d
c = d;
c->setName("carlos");
(*d).getName();
Arrays: static (stackic)

int x[5];

<table>
<thead>
<tr>
<th>loc</th>
<th>name</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Arrays: dynamic (heap)

```c
int * x;

int size = 3;
x = new int[size];

for(int i=0, i<size, i++)
    x[i] = i + 3;

delete [] x;
```