Announcements
MP2 available, due 9/16, 11:59p.

Configure your iMac 27-inch
Use the options below to build the system of your dreams

Memory
More memory (RAM) increases performance and enables your computer to perform faster and better. Choose additional 1066MHz DDR3 memory for your iMac.

The more memory your computer has, the more programs you can run simultaneously, and the better performance you get from your computer.

- Select the standard memory configuration to support day-to-day tasks such as email, word processing, and web browsing as well as more complex tasks such as editing photos, creating illustrations, and building presentations.
- Upgrade your memory to enjoy greater performance for more intensive computing tasks, such as video editing and DVD authoring.

Your iMac uses one of the fastest memory technologies available today—1066MHz, Double Data Rate (DDR3), synchronous dynamic random-access memory (SDRAM)—ensuring that the processor is constantly fed with data without wasting clock cycles.

- 4GB 1066MHz DDR3 SDRAM – 2x2GB
- 8GB 1066MHz DDR3 SDRAM – 4x2GB [Add $200.00]
- 8GB 1066MHz DDR3 SDRAM – 2x4GB [Add $600.00]
- 16GB 1066MHz DDR3 SDRAM – 4x4GB [Add $1,400.00]
Variables and memory in C++
Variables and memory in C++

<table>
<thead>
<tr>
<th>Stack memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>loc</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Pointers - Intro

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

How do we assign to `p`?

```c
p =
```

```c
p =
```

__________ operator: &

__________ operator: *

---

**Stack memory**

<table>
<thead>
<tr>
<th>loc</th>
<th>name</th>
<th>value</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>a20</td>
<td>x</td>
<td>5</td>
<td>int</td>
</tr>
<tr>
<td>a40</td>
<td>p</td>
<td></td>
<td>int *</td>
</tr>
</tbody>
</table>
Pointer variables and dynamic memory allocation:

```c
int * p;
```

### Stack memory

<table>
<thead>
<tr>
<th>loc</th>
<th>name</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p</td>
<td>int *</td>
<td></td>
</tr>
</tbody>
</table>

### Heap memory

<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>
</tbody>
</table>

Youtube: pointer binky c++
Fun and games with pointers: (warm-up)

```c
int * p, q;  // What type is q?________________
```

```c
int *p;
int x;
p = &x;
*p = 6;
cout << x;  // What is output?_____________
cout << p;  // What is output?_____________
Write a statement whose output is the value of x, using variable p: ____________
```
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:

```cpp
int * p, * q;
p = new int;
q = p;
delete p;
... // some random stuff
cout << *q;  // Do you like this?
```

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.

Use the letters S and H in a meaningful way to tell where the pointee exists in memory.
```

```
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?

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

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

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

```cpp
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;
}