



Pointers

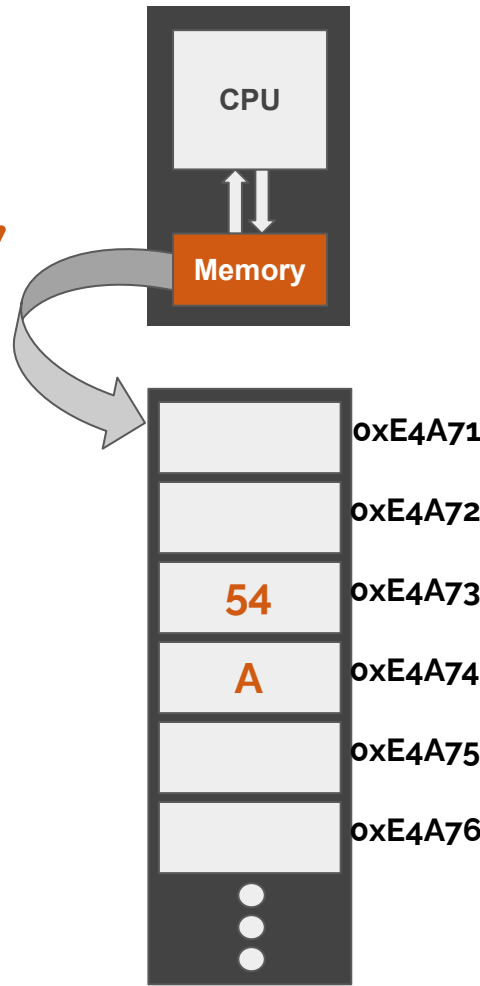


اَللّٰهُمَّ ارْزُقْنِيْ عِلْمًا نَّافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ ارْزُقْنِيْ رِزْقًا وَّاسِعًا حَلَالًا طَيِّبًا
مُّبَارَكًا مِنْ عِنْدِكَ

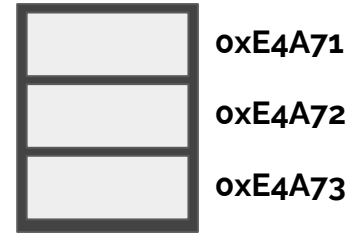
Review: Main Memory

- Memory is called **Main Memory**, **Primary Memory** or **RAM**.
- This memory is divided into **different cells**.
- Each cell has an **address** like we have address of our house numbers or PO Boxes
- CPU **stores** data into these cells and **loads** data from these cells whenever it is required.



Review: Variable Declaration

When we declare a variable, it reserves memory for a specific datatype.



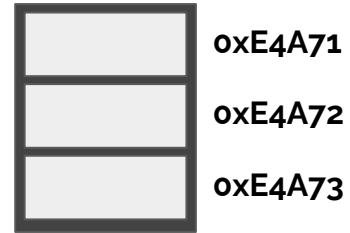
Memory

Review: Variable Declaration

When we declare a variable, it reserves memory for a specific datatype.

For Example:

```
int num;
```



Memory

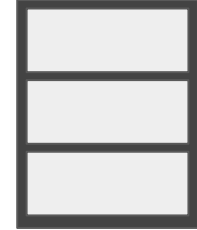
Review: Variable Declaration

When we declare a variable, it reserves memory for a specific datatype.

For Example:

```
int num;
```

num



0xE4A71

0xE4A72

0xE4A73

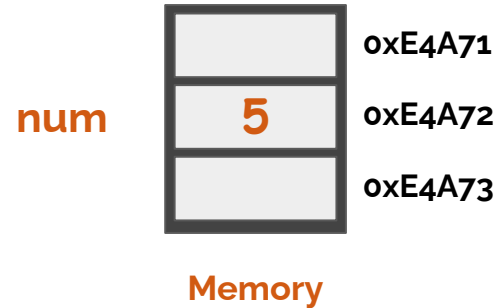
Memory

Review: Variable initialization

When we declare a variable, it reserves memory for a specific datatype.

For Example:

```
int num = 5;
```

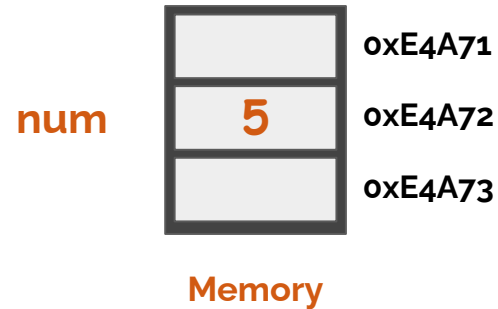


Review: Display the contents of cell

Now, if we want to display the contents of the cell then we can use the variable to see the contents of the memory.

For Example:

```
cout << num;
```



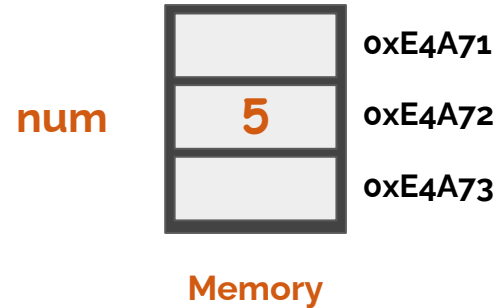
Review: Display the contents of cell

Now, if we want to display the contents of the cell then we can use the variable to see the contents of the memory.

For Example:

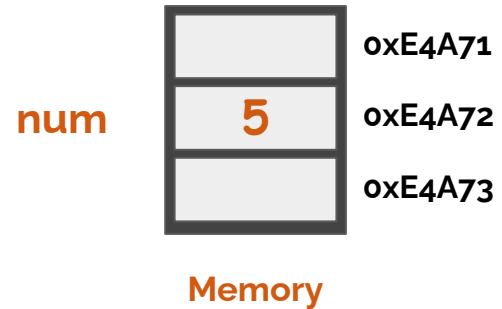
```
cout << num;
```

```
C:\C++\Week12>c++ 1.cpp -o 1.exe
C:\C++\Week12>1.exe
5
C:\C++\Week12>
```



Display the address of cell

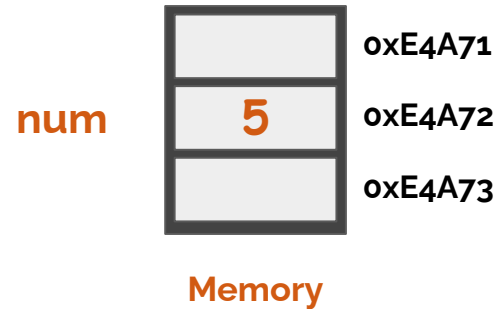
Let's say for some reason we do not want to see the **contents** (5) of the cell, but we want to see the **address** (0xE4A73) of the cell.



Display the address of cell

Let's say for some reason we do not want to see the **contents** (5) of the cell, but we want to see the **address** (0xE4A73) of the cell.

For that we write **&** before the Variable name.

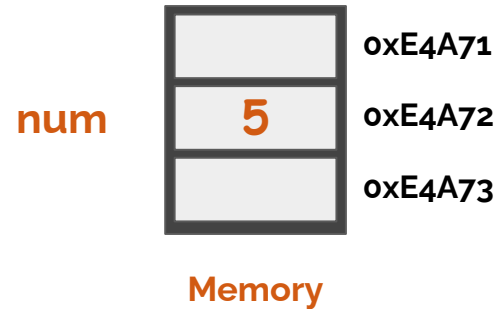


Display the address of cell

Let's say for some reason we do not want to see the **contents** (5) of the cell, but we want to see the **address** (0xE4A73) of the cell.

For that we write **&** before the Variable name.

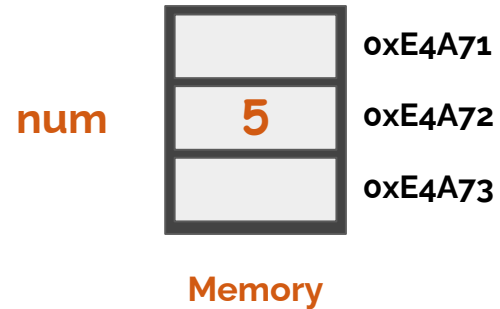
```
cout << &num;
```



Display the address of cell

Let's say for some reason we do not want to see the **contents** (5) of the cell, but we want to see the **address** (0xE4A73) of the cell.

For that we write **&** before the Variable name.



cout << #

```
C:\C++\Week12>c++ 1.cpp -o 1.exe
C:\C++\Week12>1.exe
0xE4A72
C:\C++\Week12>
```

Pointer

Before moving further, can anyone tell What is the role of a **Pointer**?



Pointer

Pointer just **points** towards something.



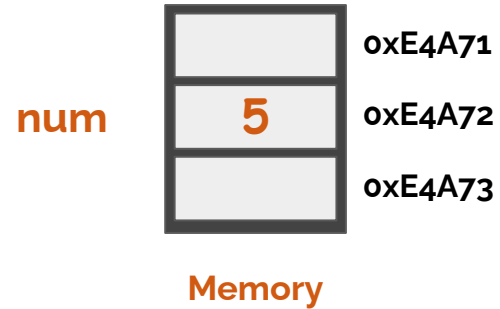
Pointer

Pointer just **points** towards something. If i aim the pointer towards the **slides** and press the button, it will start pointing towards the slides



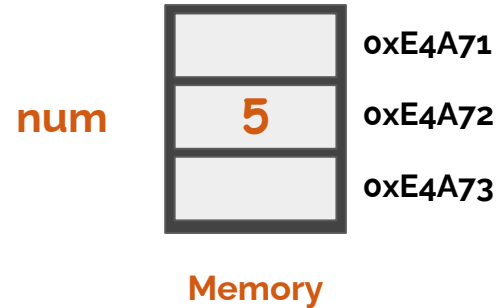
Pointers

In the same way, we have pointers in C++.



Pointers

In the same way, we have pointers in **C++**.
But We can only point towards the **address of any memory cell** using the pointer.

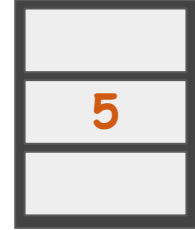


|| Pointers: Declaration

Syntax to declare a pointer of a **specific datatype** in C++ is:

```
int *p;
```

p
num



0xE4A71

0xE4A72

0xE4A73

Memory

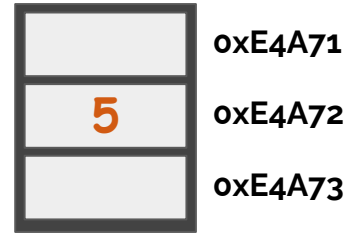
Pointers: Declaration

Syntax to declare a pointer of a **specific datatype** in C++ is:

```
int *p;
```

We have declared a pointer of **int datatype**,
It means it can only point
towards the memory cell
containing **integer** type of data.

p
num



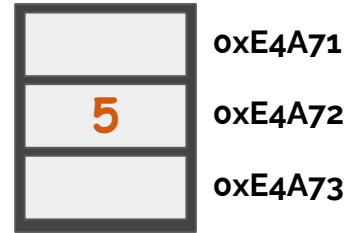
Memory

Pointers: Assign the address

Lets assign the memory address of num variable to the pointer. In short, lets press the **on** button of the pointer.

```
int *p;
```

p
num

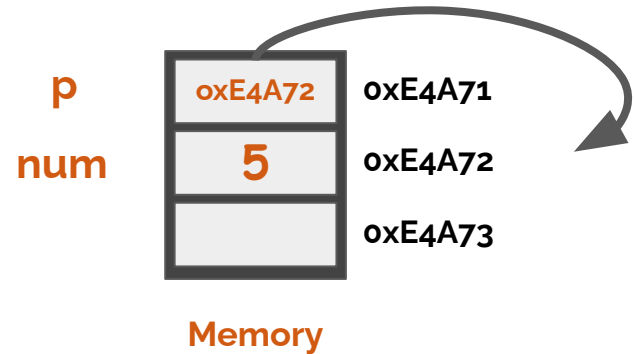


Memory

Pointers: Assign the address

Lets assign the memory address of num variable to the pointer. In short, lets press the **on** button of the pointer.

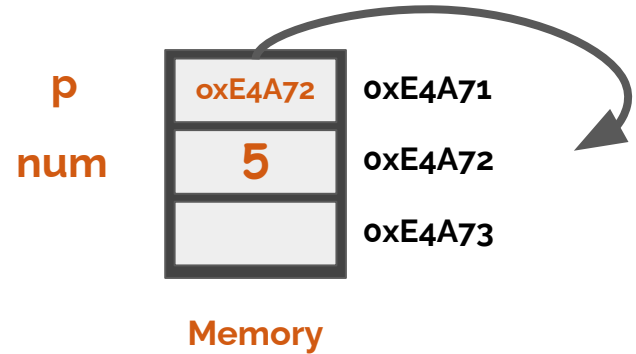
```
int *p;  
p = &num;
```



Pointers: Dereferencing

Once a pointer is assigned to an address, you can refer to the value it points to by "**dereferencing the pointer**". To do this, use the unary ***** operator

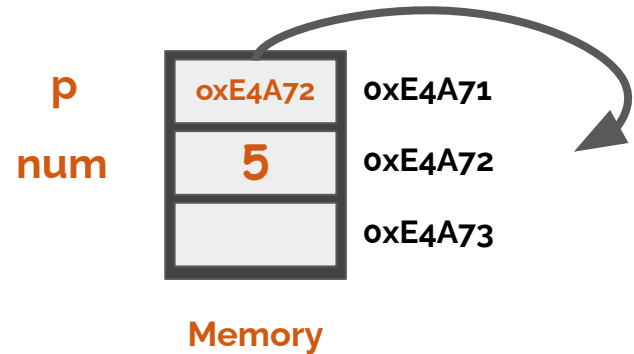
```
int *p;  
p = &num;
```



Pointers: Dereferencing

Once a pointer is assigned to an address, you can refer to the value it points to by "**dereferencing the pointer**". To do this, use the unary `*` operator

```
int *p;  
p = &num;  
cout << *p;
```



Pointers: Dereferencing

Once a pointer is assigned to an address, you can refer to the value it points to by "**dereferencing the pointer**". To do this, use the unary ***** operator

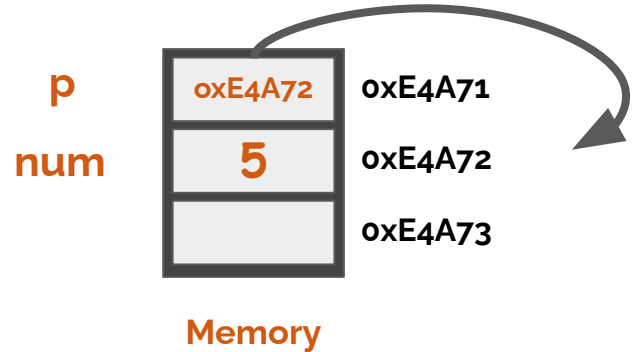
```
int *p;  
p = &num;  
cout << *p;
```

```
C:\C++\Week12>c++ 1.cpp -o 1.exe
```

```
C:\C++\Week12>1.exe
```

```
5
```

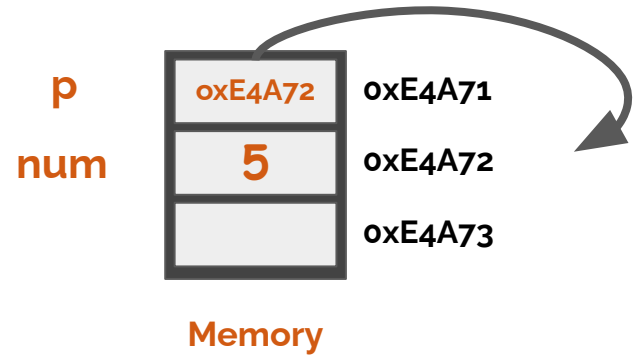
```
C:\C++\Week12>
```



Pointers: See the address to which the pointer is pointing

If we want to see which **address** the pointer is pointing, then we can write.

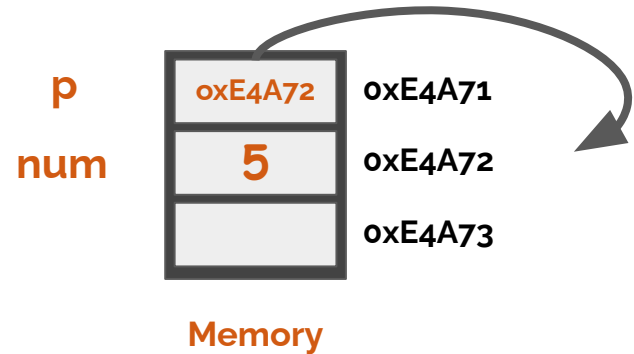
```
int *p;  
p = &num;  
cout << p;
```



Pointers: See the address to which the pointer is pointing

If we want to see which **address** the pointer is pointing, then we can write.

```
int *p;  
p = &num;  
cout << p;
```

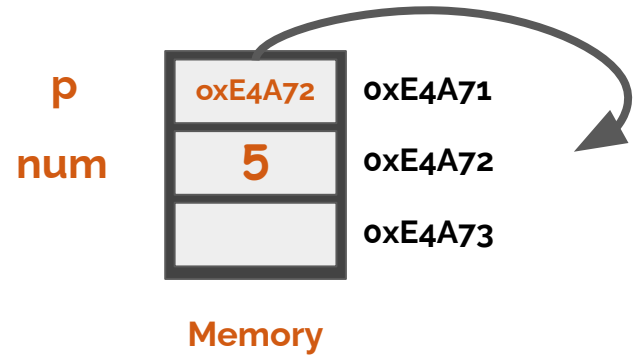


```
C:\C++\Week12>c++ 1.cpp -o 1.exe  
  
C:\C++\Week12>1.exe  
0xE4A72  
  
C:\C++\Week12>
```

Pointers: change the value of num using *p

Let's change the value of num variable using the pointer.

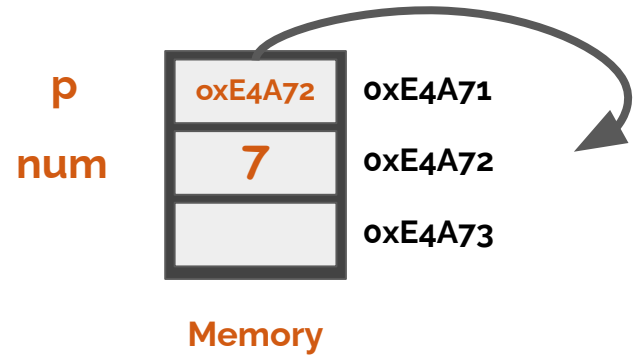
```
int *p;  
p = &num;
```



Pointers: change the value of num using *p

Let's change the value of num variable using the pointer.

```
int *p;  
p = &num;  
*p = 7;
```

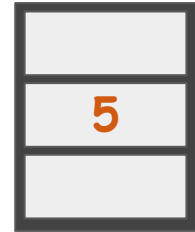


Pointers: Declaration

In the same manner we can declare pointers for any datatype

```
int *p;  
float *mypointer;  
char *another;
```

p
num



0xE4A71

0xE4A72

0xE4A73

Memory

Learning Objective

Understand the access of memory addresses using Pointer



Conclusion

- The **address** of a variable can be stored in another variable known as a **pointer variable**. The syntax for storing a variable's address to a pointer is:

```
dataType *pointerVariableName = &variableName;
```

- We need to tell the computer what the **data type** of the variable is whose address we are going to store.
- If you see the ***** in a declaration statement, a pointer is being declared for the first time.
- AFTER that, when you see the ***** on the pointer name, you are **dereferencing the pointer** to get to the target.

Self Assessment:

1. What is the output of the following code?

```
int int1 = 26;
int int2 = 45;
int *int1Ptr = &int1;
int *int2Ptr = &int2;
*int1Ptr = 89;
*int2Ptr = 62;
int1Ptr = int2Ptr;
*int1Ptr = 80;
int1 = 57;
cout << int1 << " " << int2 << endl;
cout << *int1Ptr << " " << *int2Ptr << endl;
```



Code	int1 (A1)	int2 (A2)		
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>				
<code>*int1Ptr = 89;</code>				
<code>*int2Ptr = 62;</code>				
<code>int1Ptr = int2Ptr;</code>				
<code>*int1Ptr = 80;</code>				
<code>int1 = 57;</code>				
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>				
<code>*int2Ptr = 62;</code>				
<code>int1Ptr = int2Ptr;</code>				
<code>*int1Ptr = 80;</code>				
<code>int1 = 57;</code>				
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>				
<code>int1Ptr = int2Ptr;</code>				
<code>*int1Ptr = 80;</code>				
<code>int1 = 57;</code>				
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>	89	62	A1	A2
<code>int1Ptr = int2Ptr;</code>				
<code>*int1Ptr = 80;</code>				
<code>int1 = 57;</code>				
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>	89	62	A1	A2
<code>int1Ptr = int2Ptr;</code>	89	62	A2	A2
<code>*int1Ptr = 80;</code>				
<code>int1 = 57;</code>				
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>	89	62	A1	A2
<code>int1Ptr = int2Ptr;</code>	89	62	A2	A2
<code>*int1Ptr = 80;</code>	89	80	A2	A2
<code>int1 = 57;</code>				
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>	89	62	A1	A2
<code>int1Ptr = int2Ptr;</code>	89	62	A2	A2
<code>*int1Ptr = 80;</code>	89	80	A2	A2
<code>int1 = 57;</code>	57	80	A2	A2
<code>cout << int1 << " " << int2 << endl;</code>				
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>	89	62	A1	A2
<code>int1Ptr = int2Ptr;</code>	89	62	A2	A2
<code>*int1Ptr = 80;</code>	89	80	A2	A2
<code>int1 = 57;</code>	57	80	A2	A2
<code>cout << int1 << " " << int2 << endl;</code>	57 80			
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>				

Code	int1 (A1)	int2 (A2)	int1Ptr	int2Ptr
<code>int int1 = 26;</code> <code>int int2 = 45;</code>	26	45		
<code>int *int1Ptr = &int1;</code> <code>int *int2Ptr = &int2;</code>	26	45	A1	A2
<code>*int1Ptr = 89;</code>	89	45	A1	A2
<code>*int2Ptr = 62;</code>	89	62	A1	A2
<code>int1Ptr = int2Ptr;</code>	89	62	A2	A2
<code>*int1Ptr = 80;</code>	89	80	A2	A2
<code>int1 = 57;</code>	57	80	A2	A2
<code>cout << int1 << " " << int2 << endl;</code>	57 80			
<code>cout << *int1Ptr << " " << *int2Ptr << endl;</code>	80 80			

Self Assessment:

2. What is the output of the following code?

```
string str1 = "sunny";  
string str2 = "cloudy";  
string *s1;  
cout << str1 << " " << str2 << endl;  
s1 = &str1;  
str1 = str2;  
str2 = *s1;  
cout << str1 << " " << str2 << endl;
```



Code	str1 (A1)	str2 (A2)	
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;			
cout << str1 << " " << str2 << endl;			
s1 = &str1;			
str1 = str2;			
str2 = *s1;			
cout << str1 << " " << str2 << endl;			

Code	str1 (A1)	str2 (A2)	s1
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;	sunny	cloudy	
cout << str1 << " " << str2 << endl;			
s1 = &str1;			
str1 = str2;			
str2 = *s1;			
cout << str1 << " " << str2 << endl;			

Code	str1 (A1)	str2 (A2)	s1
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;	sunny	cloudy	
cout << str1 << " " << str2 << endl;	sunny cloudy		
s1 = &str1;			
str1 = str2;			
str2 = *s1;			
cout << str1 << " " << str2 << endl;			

Code	str1 (A1)	str2 (A2)	s1
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;	sunny	cloudy	
cout << str1 << " " << str2 << endl;	sunny cloudy		
s1 = &str1;	sunny	cloudy	A1
str1 = str2;			
str2 = *s1;			
cout << str1 << " " << str2 << endl;			

Code	str1 (A1)	str2 (A2)	s1
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;	sunny	cloudy	
cout << str1 << " " << str2 << endl;	sunny cloudy		
s1 = &str1;	sunny	cloudy	A1
str1 = str2;	cloudy	cloudy	A1
str2 = *s1;			
cout << str1 << " " << str2 << endl;			

Code	str1 (A1)	str2 (A2)	s1
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;	sunny	cloudy	
cout << str1 << " " << str2 << endl;	sunny cloudy		
s1 = &str1;	sunny	cloudy	A1
str1 = str2;	cloudy	cloudy	A1
str2 = *s1;	cloudy	cloudy	A1
cout << str1 << " " << str2 << endl;			

Code	str1 (A1)	str2 (A2)	s1
string str1 = "sunny"; string str2 = "cloudy";	sunny	cloudy	
string *s1;	sunny	cloudy	
cout << str1 << " " << str2 << endl;	sunny cloudy		
s1 = &str1;	sunny	cloudy	A1
str1 = str2;	cloudy	cloudy	A1
str2 = *s1;	cloudy	cloudy	A1
cout << str1 << " " << str2 << endl;	cloudy cloudy		

Self Assessment:

3. What is the output of the following code?

```
double dec1 = 2.5;
double dec2 = 3.8;
double *p, *q;
p = &dec1;
*p = dec2 - dec1;
q = p;
*q = 10.0;
*p = 2 * dec1 + (*q);
q = &dec2;
dec1 = *p + *q;
cout << dec1 << " " << dec2 << endl;
cout << *p << " " << *q << endl;
```



Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;				
*p = dec2 - dec1;				
q = p;				
*q = 10.0;				
*p = 2 * dec1 + (*q);				
q = &dec2;				
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;				
q = p;				
*q = 10.0;				
*p = 2 * dec1 + (*q);				
q = &dec2;				
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;				
*q = 10.0;				
*p = 2 * dec1 + (*q);				
q = &dec2;				
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;	1.3	3.8	A1	A1
*q = 10.0;				
*p = 2 * dec1 + (*q);				
q = &dec2;				
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;	1.3	3.8	A1	A1
*q = 10.0;	10	3.8	A1	A1
*p = 2 * dec1 + (*q);				
q = &dec2;				
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;	1.3	3.8	A1	A1
*q = 10.0;	10	3.8	A1	A1
*p = 2 * dec1 + (*q);	30	3.8	A1	A1
q = &dec2;				
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;	1.3	3.8	A1	A1
*q = 10.0;	10	3.8	A1	A1
*p = 2 * dec1 + (*q);	30	3.8	A1	A1
q = &dec2;	30	3.8	A1	A2
dec1 = *p + *q;				
cout << dec1 << " " << dec2 << endl;				
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
<pre>double dec1 = 2.5; double dec2 = 3.8; double *p, *q;</pre>	2.5	3.8		
<pre>p = &dec1;</pre>	2.5	3.8	A1	
<pre>*p = dec2 - dec1;</pre>	1.3	3.8	A1	
<pre>q = p;</pre>	1.3	3.8	A1	A1
<pre>*q = 10.0;</pre>	10	3.8	A1	A1
<pre>*p = 2 * dec1 + (*q);</pre>	30	3.8	A1	A1
<pre>q = &dec2;</pre>	30	3.8	A1	A2
<pre>dec1 = *p + *q;</pre>	33.8	3.8	A1	A2
<pre>cout << dec1 << " " << dec2 << endl;</pre>				
<pre>cout << *p << " " << *q << endl;</pre>				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;	1.3	3.8	A1	A1
*q = 10.0;	10	3.8	A1	A1
*p = 2 * dec1 + (*q);	30	3.8	A1	A1
q = &dec2;	30	3.8	A1	A2
dec1 = *p + *q;	33.8	3.8	A1	A2
cout << dec1 << " " << dec2 << endl;	33.8 3.8			
cout << *p << " " << *q << endl;				

Code	dec1 (A1)	dec2 (A2)	p	q
double dec1 = 2.5; double dec2 = 3.8; double *p, *q;	2.5	3.8		
p = &dec1;	2.5	3.8	A1	
*p = dec2 - dec1;	1.3	3.8	A1	
q = p;	1.3	3.8	A1	A1
*q = 10.0;	10	3.8	A1	A1
*p = 2 * dec1 + (*q);	30	3.8	A1	A1
q = &dec2;	30	3.8	A1	A2
dec1 = *p + *q;	33.8	3.8	A1	A2
cout << dec1 << " " << dec2 << endl;	33.8 3.8			
cout << *p << " " << *q << endl;	33.8 3.8			