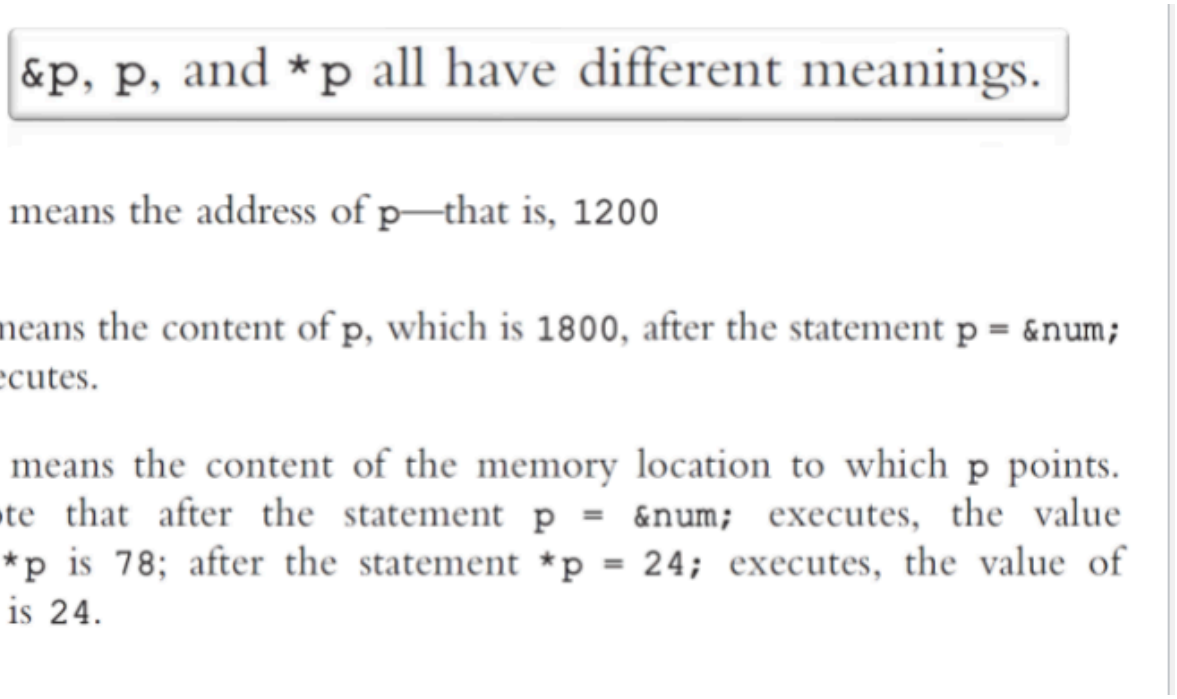
*Basic Pointers:



&p, p, and *p all have different meanings.

&p means the address of p—that is, 1200

p means the content of p, which is 1800, after the statement p = # executes.

*p means the content of the memory location to which p points. Note that after the statement p = # executes, the value of *p is 78; after the statement *p = 24; executes, the value of *p is 24.

P -If we were to cout P it would output the memory address to the variable it points to. Remember it holds the memory address of the variable it points to
*P - If were to cout *P, it would output the what memory address holds

1) **What does it mean to allocate memory?:**
- Basically, in your computer there is an unused slot of memory in your ram. By using the new operator, we can assign what this unused slot of memory should hold. The new operator returns a address, that's why we use a pointer to store it.
- So what are the types of allocated memory we can have?: This is allocated memory of a variable, a array, and a class type

2) **When do we deallocate memory?:**
- When we no longer need memory, we can deallocate it with the help of the delete operator and destructor
- The destructor is called when we use the delete operator or when a object goes out of scope
- **After you use the delete operator, you should set the pointer to null so it does not become a dangling pointer.**

What's the deal with the "new" and "delete" operator

1) **C++ Uses what to create "dynamic Variables?":**

Pointers

2) **What are the operators we can use to create and destroy these dynamic variables?:**

* The new operator!:
  - The new operator allocates memory for the given type and returns a pointer for the given type.
  - Apart of this syntax is used to create dynamic variables
  - **Let's illustrate this:**
  - **Please note that the below demonstrates how to dynamically allocate variables, very similar to dynamically allocating class objects**

```
int *p;           //p is a pointer of type int
char *name;       //name is a pointer of type char
string *str;      //str is a pointer of type string

p = new int;      //allocates memory of type int
                  //and stores the address of the
                  //allocated memory in p
*p = 28;          //stores 28 in the allocated memory
```

P, a variable, is now dynamically allocated

```
int main()
{
    int *x; //declareda a pointer of a integer typed
    x = new int; //the new operator allocates memory for an int type and the pointer variable holds the memeory address of the allo
    cout << x; //the pointer points to the allocated memeory (holds memeory address of the allocated memeory)
    cout << endl;
    cout << new int; //this will cout the allocated memeory, not is contents but the memeory address
    cout << endl; //endl

    *x = 58; //this will change the contents of the allocated memeory

    cout << *x; //this will cout the contents of the allocated memory

    return 0;
```

```
name = new char[5];     //allocates memory for an array of
                        //five components of type char and
                        //stores the base address of the array
                        //in name
strcpy(name, "John");   //stores John in name
```

1) A pointer to a class object:
   - A pointer for a class object type

2) A pointer to an integer:
   - A pointer of an integer type

3) A pointer of a string type:
   - A pointer to a string

4) A pointer of a character type:
   - A pointer to a character variable

“Dynamicaly allocates memory a for an array of char type”
“Class holds a data array of char chars dynamically allocated to a pointer.”

```
#include <iostream>
using namespace std;

class Point {
public:
   Point(double xValue = 0, double yValue = 0);
   void Print();

   double X;
   double Y;
};

Point:: Point(double xValue, double yValue) {
   X = xValue;
   Y = yValue;
}

void Point::Print() {
   cout << "(" << X << ", ";
   cout << Y << ")" << endl;
}

int main() {
   Point* point1 = new Point;
   (*point1).Print();

   Point* point2 = new Point(8, 9);
   (*point2).Print();

   return 0;
}
```

The above dynamically allocates memory for a classname **object**, syntax is similar to dynamically allocating variables

1) **What does the syntax mean in**
   Point* point1 = new Point**;**
   We created a pointer to a class object, I.E. a pointer of class object type
   What follows is the new operator followed by the class name. It dynamically allocates memory for the class object type and the pointer is pointing to this. Meaning that the pointer not only holds the memory address to which this dynamically allocated memory is stored at, but derefercing this pointer will gives us access to the contents of the allocated memory. Which is basically the the class object's members.
   The new operator dynamically allocates memory for the class type

2) **Is the pointer the object?:**
   - Nope, the pointer here is not acting as the object for the class. The object is still created, but it is created dynamically on the heap rather than on the stack. The pointer stores the memory address of the dynamically allocated object.

3) **When it comes to accessing members from the class, it is said that we use the pointer along with the member access operator "to access the members of the OBJECT." Why is it called object instead of class?:**
   - When I mentioned "object" in that context, I was referring to the instance of the class. In C++, we often refer to instances of classes as objects. So when I said "we use the pointer to access the methods and attributes of the object", I meant that we use the pointer to access the methods and attributes of the instance of the class that the pointer is pointing to