POINTER OPERATORS

Working with pointers and variable addresses

Delroy A. Brinkerhoff

IMPORTANT OPERATOR CONCEPTS

- There are a limited number of characters on the keyboard, forcing computer languages to reuse some characters
- Operators that have multiple meanings are said to be overloaded
- Overloaded operators whose meaning depends on where they are used are said to be context sensitive
- As you study the pointer operators, take note of
  - The symbol or characters forming each operator
  - Where the operators are used
  - The meaning and behavior of each operator, which is often tied to the operator's name

POINTER OPERATORS

Operator	Name	Example

`*`	Pointer Definition	`int*     i;`
		`Person*  pptr;`

`*`	Dereference, Indirection	`*i = 123;`
		`cout << *int_ptr << endl;`

`&`	Address of	`Person p;`
		`pptr = &p;`

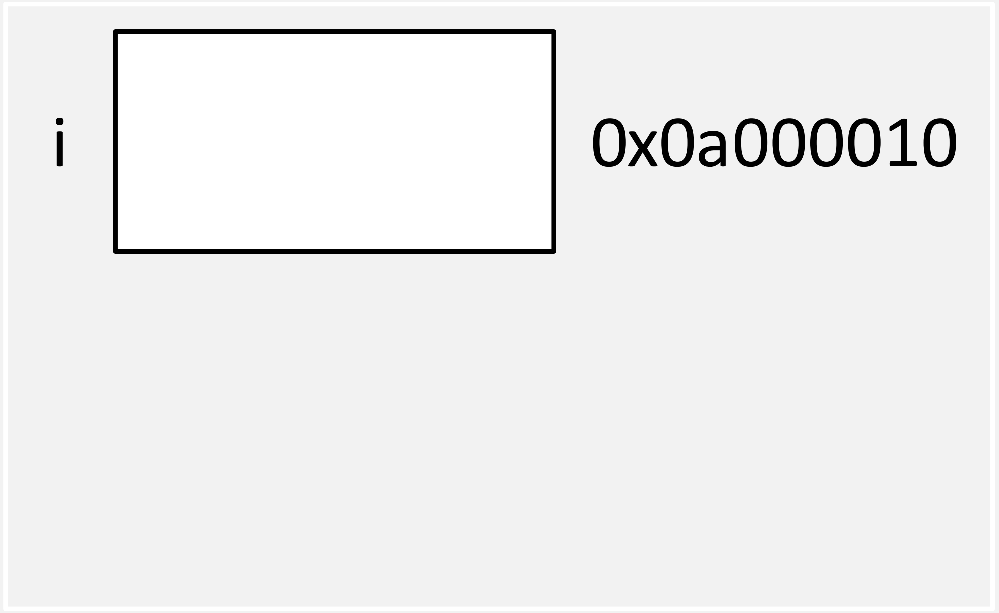
`new`	New	`pptr = new Person;`

`delete`	Delete	`delete p;`
		`delete pptr;`

`->`	Arrow	`cout << pptr->name << endl;`

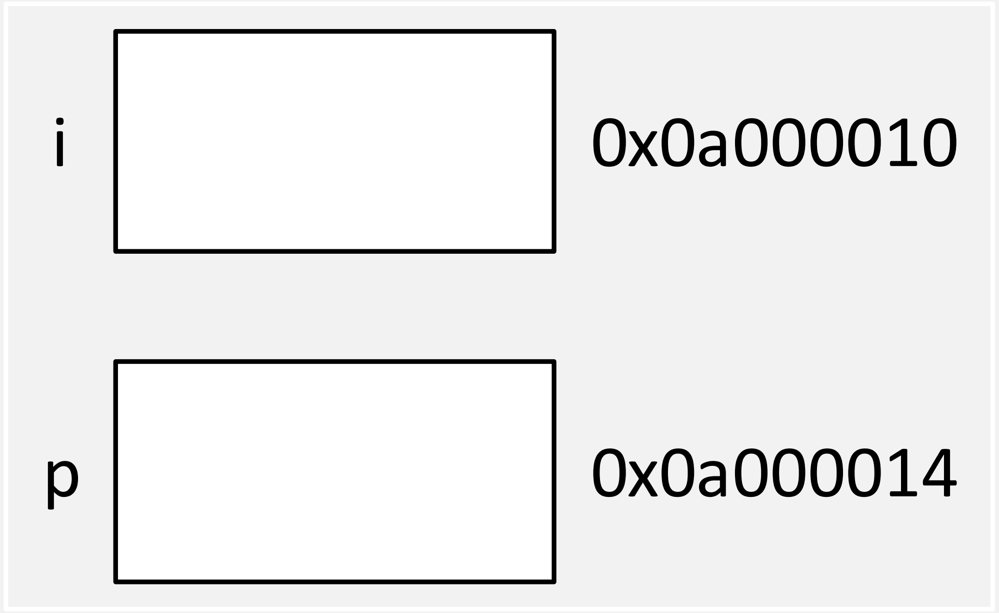
POINTER OPERATOR EXAMPLES

```
int     i;
```



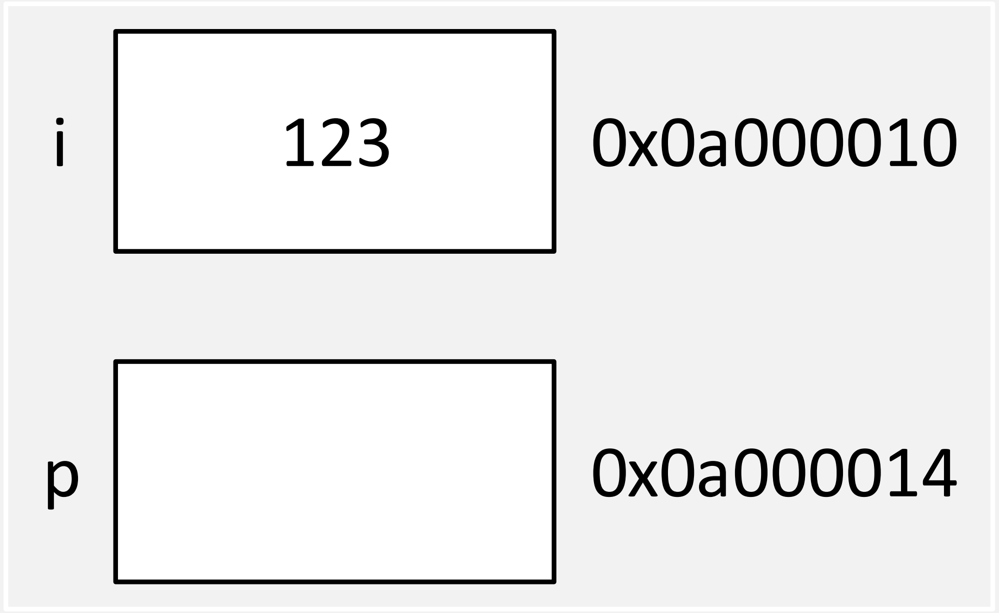
POINTER OPERATOR EXAMPLES

```
int    i;
int*   p;
```



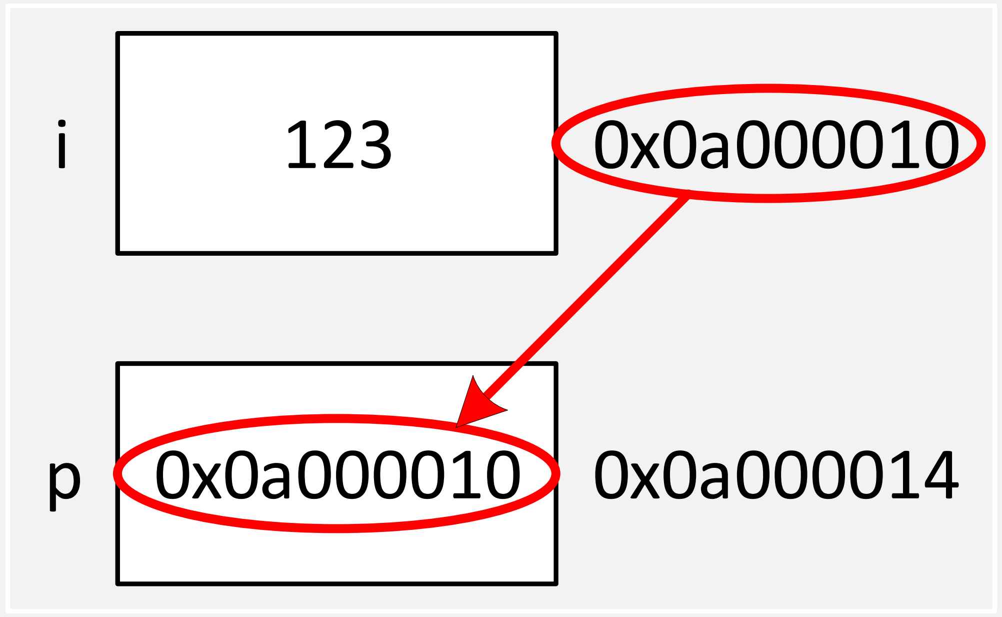
POINTER OPERATOR EXAMPLES

```
int    i;
int*   p;

i = 123;
```



POINTER OPERATOR EXAMPLES

```
int    i;
int*   p;

i = 123;
p = &i;
```



i
123
0x0a000010
p
0x0a000010
0x0a000014