‘*’ , ‘&’ → Pointers

Data Structure

Assist. Prof. Dr. Abdul Hadi Mohammed

What is a Pointer ?

POINTERS

Point to here, point to there, point to that, point to this, and point to nothing!
well, they are just memory addresses!!??

- In a generic sense, a “pointer” is anything that tells us where something can be found.
  - Addresses in the phone book
  - URLs for webpages
  - Road signs

C Pointer Variables

To declare a pointer variable, we must do two things

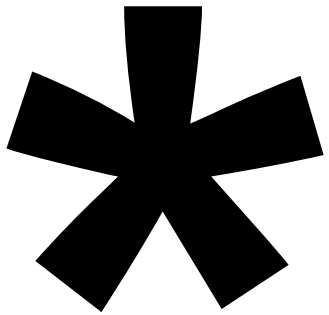
- Use the "*" (star) character to indicate that the variable being defined is a pointer type.
- Indicate the type of variable to which the pointer will point (the pointee). This is necessary because C provides operations on pointers (e.g., *, ++, etc) whose meaning depends on the type of the pointee.

- General declaration of a pointer

```
type *nameOfPointer;
```

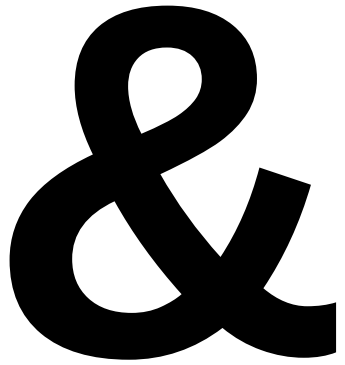
Operators used in Pointers

Dereferencing



(Value of)

Address



(Address of)

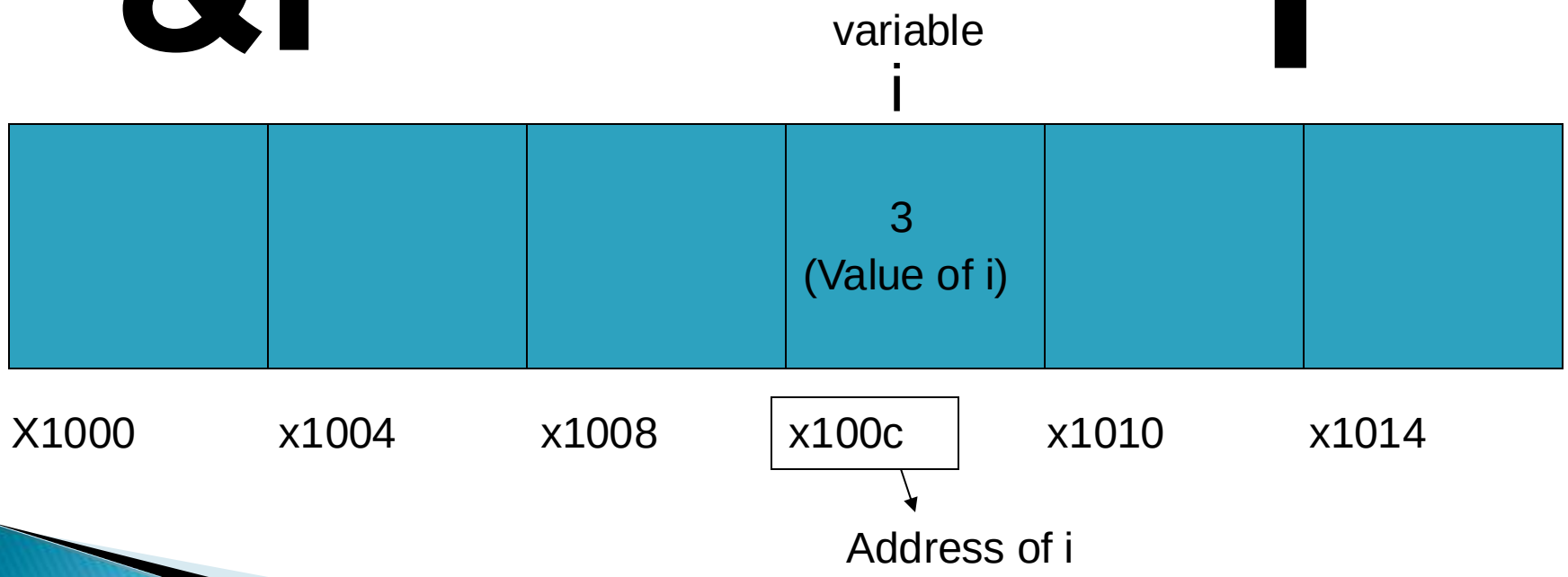
Int i=3;

Address of ‘i’

‘&i’

Value of ‘i’

‘*i’



The value ‘3’ is saved in the memory location ‘x100c’

Syntax for pointers

(pointer type declaration)

*type *identifier ;*

Example

Char *cp ;

Int *ip ;

Double *dp ;

Pointer Assignment

```
Int i = 1 , *ip ; //pointer declaration

ip = &i ; //pointer assignment

*ip = 3 ;  //pointer assignment
```

POINTERS

- Other pointer declarations that you may find and can make you confused are listed below.

Pointer declaration	Description
`int *x`	`x` is a pointer to `int` data type.
`int *x[10]`	`x` is an `array[10]` of pointer to `int` data type.
`int *(x[10])`	`x` is an `array[10]` of pointer to `int` data type.
`int **x`	`x` is a pointer to a pointer to an `int` data type – double pointers.
`int (*x)[10]`	`x` is a pointer to an `array[10]` of `int` data type.
`int *funct()`	`funct()` is a function returning an integer pointer.
`int (*funct)()`	`funct()` is a pointer to a function returning `int` data type – quite familiar constructs.
`int (*(*funct())[10])()`	`funct()` is a function returning pointer to an `array[10]` of pointers to functions returning `int`.
`int (*(*x[4])())[5]`	`x` is an `array[4]` of pointers to functions returning pointers to `array[5]` of `int`.

Pointer Arithmetic

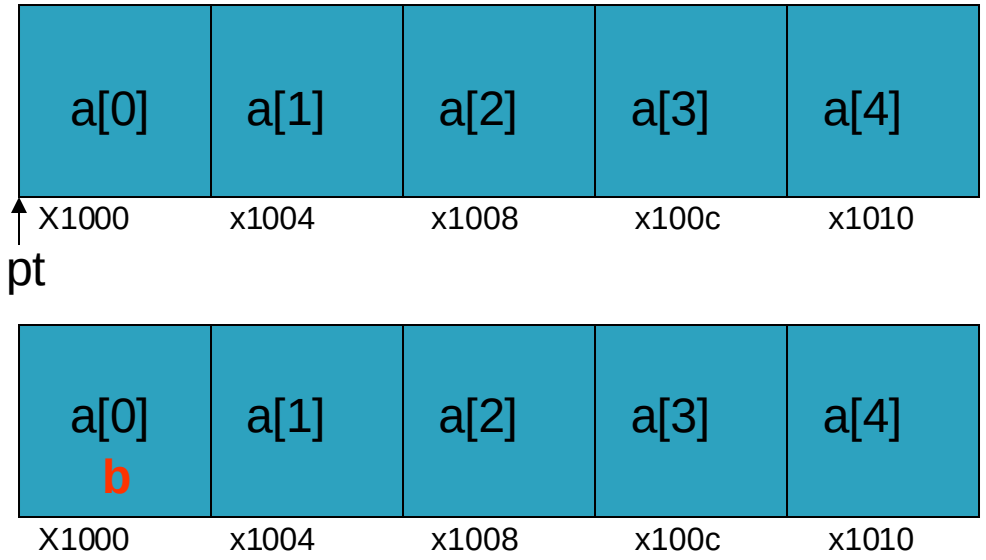
Lets take this example program

```
#include<stdio.h>
Void main()
{
Int a [5]={1,2,3,4,5} , b , *pt ;
pt = &a[0];
pt = pt + 4 ;
b=a[0] ;
b+=4 ;
}
```

b = 1

b=1+4

b= 5



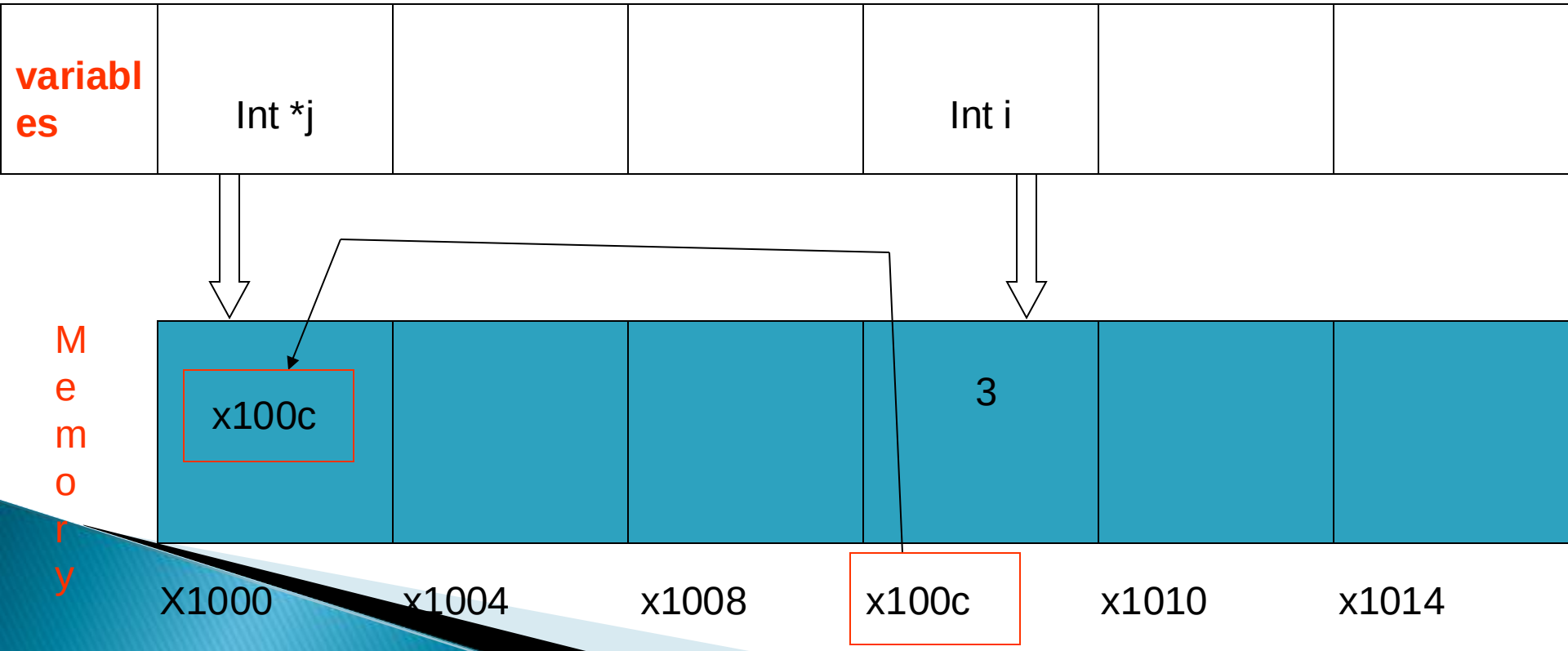
Lets Take an Example and See how pointers work

```
#include<stdio.h>
Void main()
{
Int i=3;
Int *j;
j=&i;
Printf(“i=%d”i);
Printf(“*j=%d”*j);
}
```

Int i=3; Create an integer variable ‘i’ and initialize it to 3

Int *j; Create a pointer variable ‘j’- create value of ‘j’

j = &i; Initialize the pointer value of ‘j’ to the address of ‘i’



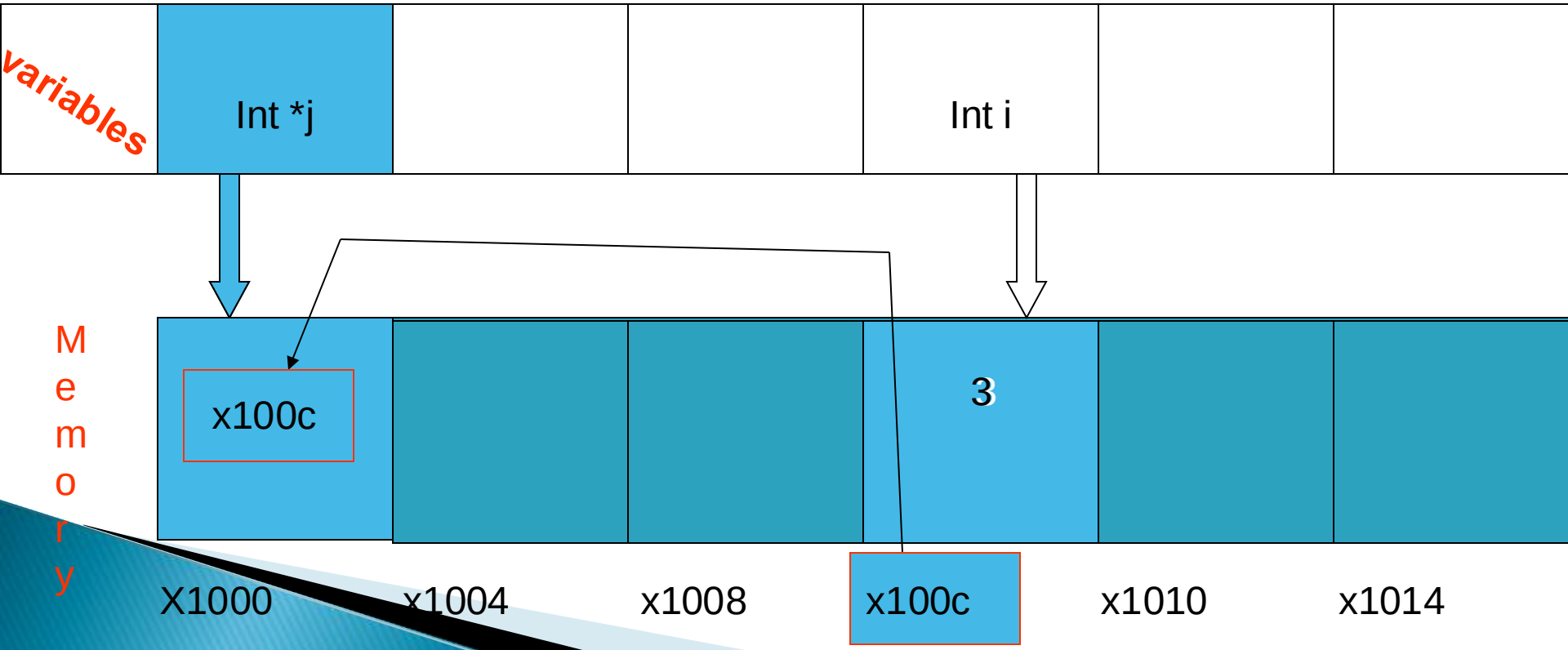
Printf(“i=%d” i);
Printf(“*j=%d” *j);

Output screen

We know j=&i

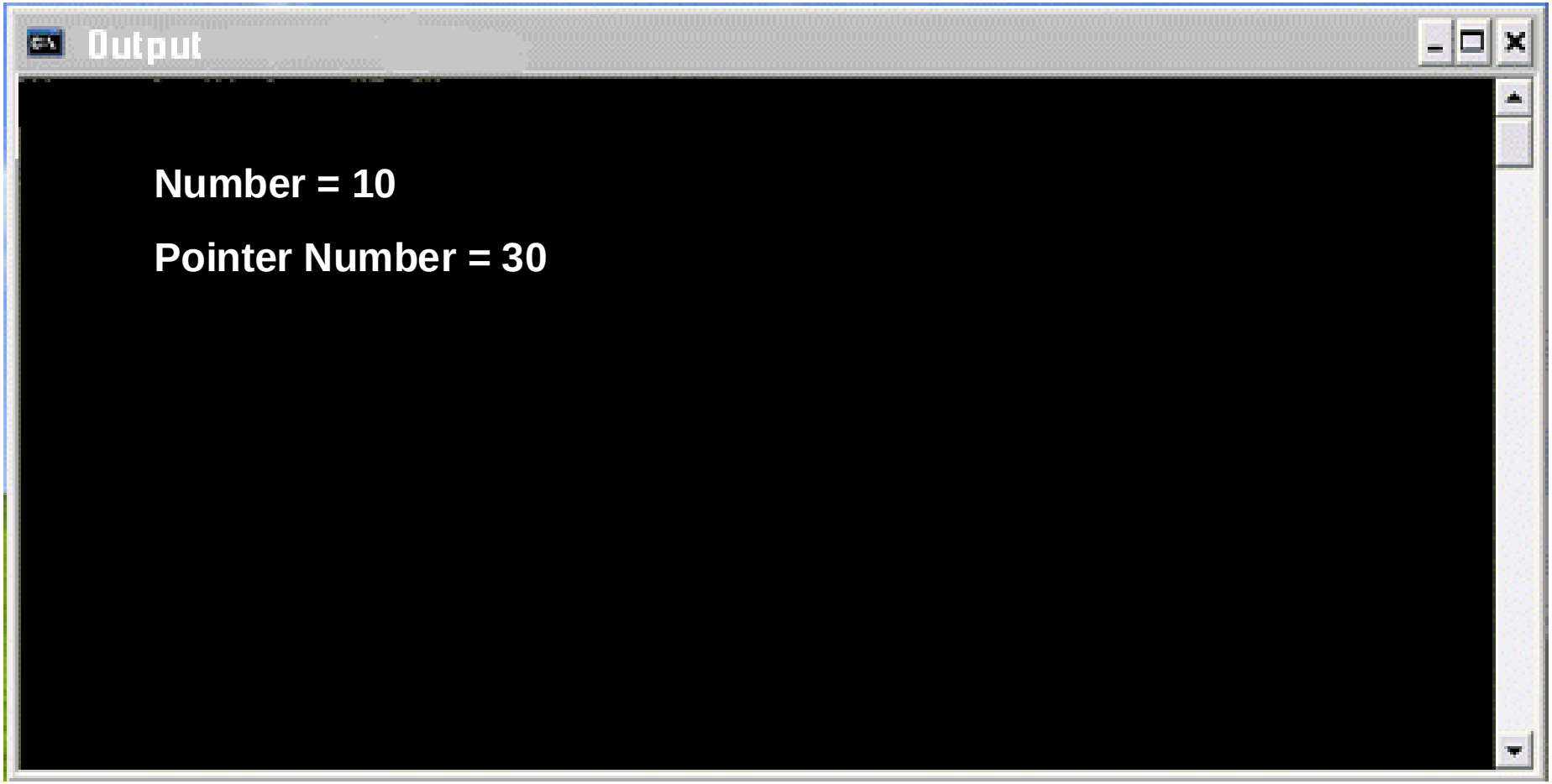
So → *j=*(&i) value of (address of i)

(i.e.) value in address (x100c)



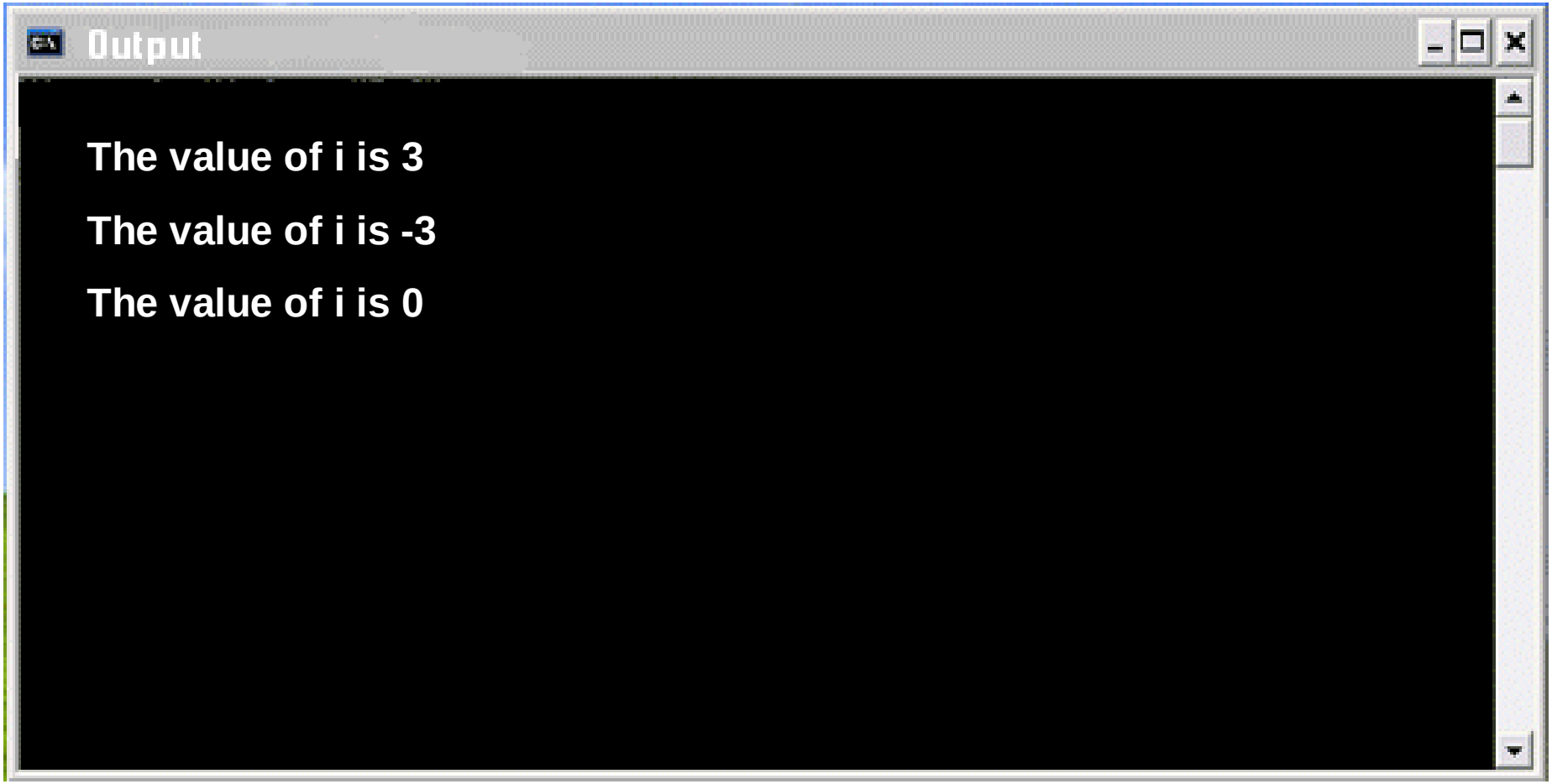
Predict the output of this code

```
int main()
{
    int num = 10;
    int* pnum = nullptr;  // Use nullptr instead of
NULL in modern C++
    pnum = &num;
    *pnum += 20;
    cout << "\nNumber = " << num << endl;
    cout << "Pointer Number = " << *pnum << endl;
    return 0;
}
```


Output
Number = 10
Pointer Number = 30

Work to your Brain

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 12};
int *p, *q, i;
p = &a[2];
q = &a[5];
i = *q - *p;
cout << "The value of i is " << i << endl;
i = *p - *q;
cout << "The value of i is " << i << endl;
a[2] = a[5] = 0;
cout << "The value of i is " << i << endl;
```


Output
The value of i is 3
The value of i is -3
The value of i is 0

Work to your Brain

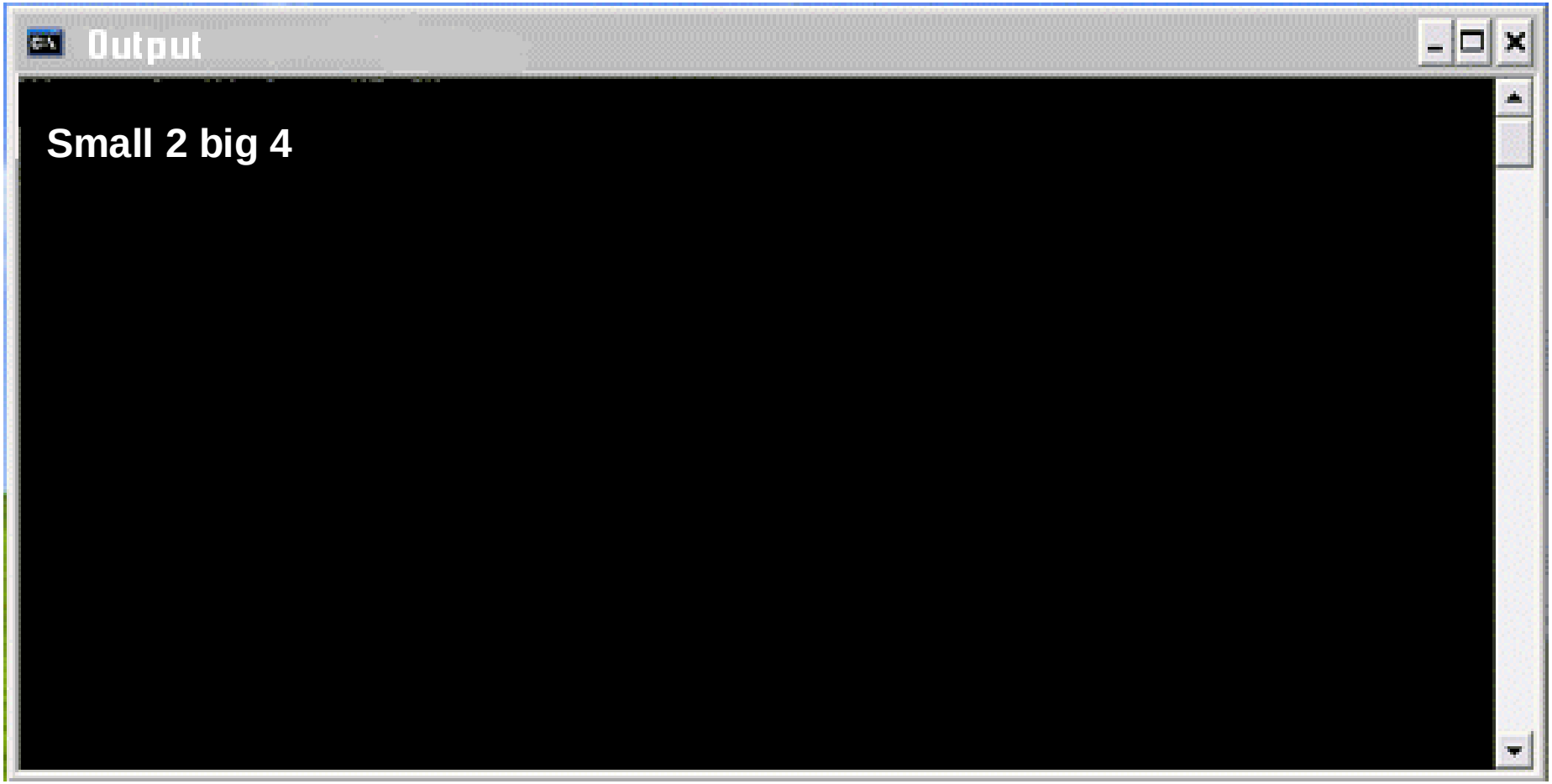
```
int a[10] = { 2,3,4,5,6,7,8,9,1,0 }, *p, *q;
p = &a[2];
q = p + 3;
p = q – 1;
p+ + ;
cout << "The values of p and q are: " << *p << ", " <<
*q << endl;
```


Output
The value of p and q are : 7 , 7

Work to your Brain

```
int main() {
int x[2] = {1, 2}, y[2] = {3, 4};
int small,big;
small = &x[0];
big = &y[0];
min_max(&small,&big);
cout << "small: " << *small << " big:
" << *big << endl;
return 0;
}
```

```
min_max(int *a,int *b)
{
a++;
b++;
return (*a,*b);
}
```


Output
Small 2 big 4