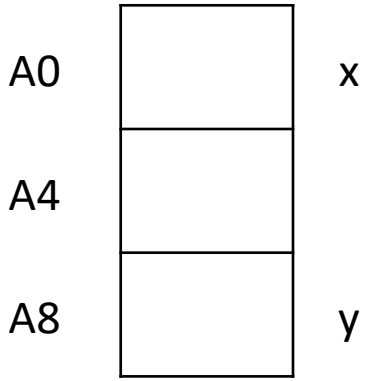
Pointers

Storing Variables

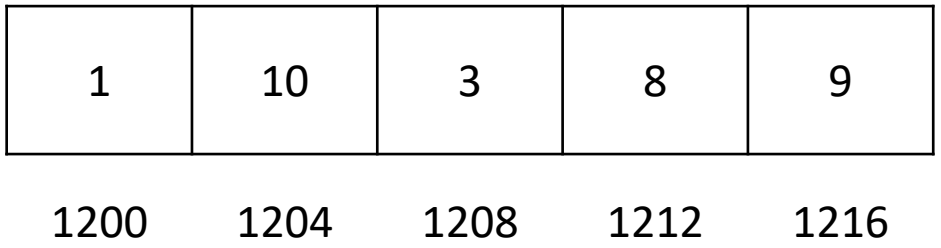
- Each variable is stored in a particular address (memory location):

```
int x=8;
int y=3;
```



Array Storage

int values[5] = {1,10,3,8,9};



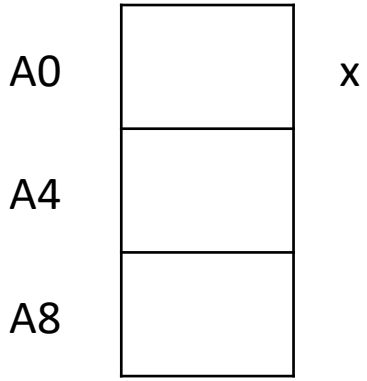
Pointer

- A ‘pointer’ is a variable that can be set to point to any of these memory addresses
- And it can be changed to point to a different memory address when needed
  - So a pointer could first point to the memory address of the variable ‘x’ and then to the memory address of the variable ‘y’, etc.
- Or a pointer to point to the memory address of the first element of an array, then to the second element, then to the third, etc.

Accessing a Memory Address

- Suppose that the following variable is declared:

```
int x;   // Suppose it's memory address is A0
x = 8;
```



- '&' is the "address"

```
cout << &x;  // Displays ?
```

Declaring a Pointer

- To declare a pointer variable:

```
int* ptr;
```

OR

```
int *ptr;
```

OR

```
int * ptr;
```

- The variable ‘ptr’ can be set to point to any integer variable, or any element of an ‘int’ array

Initialising a Pointer

- To initialize an ‘int’ variable:

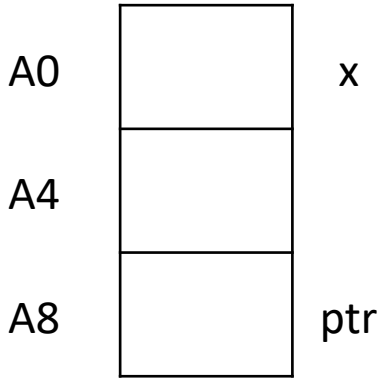
```
int x=0;
```

- To initialize a pointer variable:

```
int* ptr = nullptr;
```

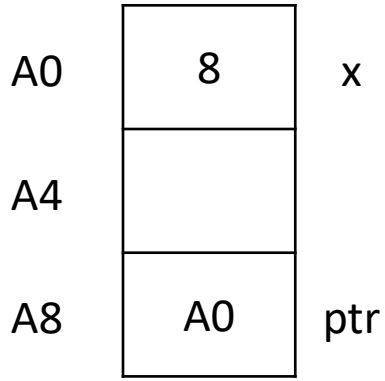
Setting Pointers

```
int x=8;
int* ptr = nullptr;

ptr = &x;
```



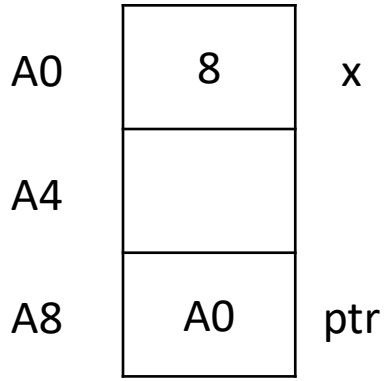
Setting a Pointer

```
int x=8;
int* ptr = nullptr;


ptr = &x;


cout << *ptr;          // “Value at the location to which ptr points”
```



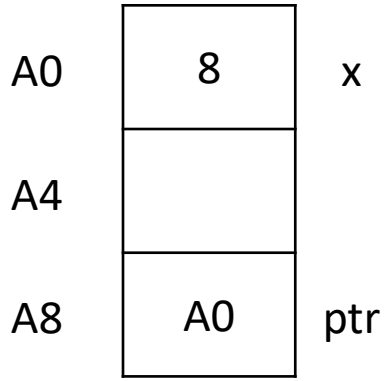
Setting a Pointer

```
int x=8;
int* ptr = nullptr;

ptr = &x;

cout << x;
cout << &x;
cout << ptr;
cout << &ptr;
cout << *ptr;
```



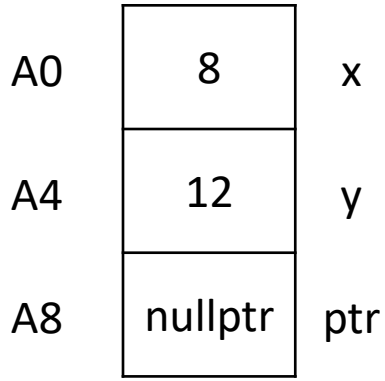
Using a Pointer

```
int x=8;
int* ptr = nullptr;


ptr = &x;


*ptr = 10;      // “Value at the location to which ptr points is
                //    assigned 10”
```



Using a Pointer

```
int x=8;
int y=12;
int* ptr = nullptr;

ptr = &y;
cout << ptr;
cout << *ptr;
```



Using a Pointer

```
int x=8;
int y=12;
int* ptr = nullptr;

ptr = &y;
*ptr += 5;
cout << y;

ptr = &x;
x = 20;
cout << *ptr;
```