# Array

Prepared by: Lec Tasmiah Tamzid Anannya, CS Dept, AIUB

# Practice Problems

- Write a program that reads ten numbers from the user and checks if any of them match.

- Write a program that checks password for a system. Suppose the password is "TROJAN". Now ask the user for the password, if the user's password matches show a message, "Successful", else show, "Not successful."

- Declare two integer array of size 10. Now keep the addition of these two array into another array and dislay the value of the result array.

# Pointer

# Address of a variable

- Before you get into the concept of pointers, let's first get familiar with address in C.
- If you have variable called var, &var will give the address of this variable in the memory.
- And it is called reference operator.

# Example

```
#include <stdio.h>
int main()
{
  int var = 5;
  printf("Value: %d\n", var);
  printf("Address: %u", &var);  //Notice, the ampersand(&) before var.
  return 0;
}
```

# Example

```
#include <stdio.h>
int main()
{
  int var = 5;
  printf("Value: %d\n", var);
  printf("Address: %u", &var);  //Notice, the ampersand(&) before var.
  return 0;
}
```

```
Value: 5
Address: 2686778
```

# Example

```
#include <stdio.h>
int main()
{
  int var = 5;
  printf("Value: %d\n", var);
  printf("Address: %u", &var);  //Notice, the ampersand(&) before var.
  return 0;
}
```

```
Value: 5
Address: 2686778
```

In above source code, value 5 is stored in the memory location 2686778.
var is just given a name to that location.

# Pointer variable

- In C, you can create a special variable that stores the address (rather than the value). This variable is called pointer variable or simply a pointer.
- For example,

**data_type *variablename;**

**int *p;**

- Above statement defines, p as an pointer variable of type int.

# Reference operator (&) and Dereference operator (*)

- & is called the reference variable and gives the address of a variable.
- Likewise, there is another operator that gets you the value from the address, it is called a dereference operator *.

# Example

```
#include <stdio.h>
int main()
{
  int *p,q;
  q=100;
  p=&q;
  printf("%d %d\n %d %d", *p, p, q, &q);
}
```

# Example

```
#include <stdio.h>
int main()
{
  int *p,q;
  q=100;
  p=&q;
  printf("%d %d\n %d %d", *p, p, q, &q);
}
```

```
100 6356744
100 6356744
```

# Let's see it graphically

1.

```
int *p, q;
```

Suppose, q is located at memory address 6356744 and p is right before it, at location 6356742 .

# Let's see it graphically

1.

int *p, q;

| Location | Contents | |
|---|---|---|
| 6356742 | | location of p |
| 6356744 | | location of q |

Suppose, q is located at memory address 6356744 and p is right before it, at location 6356742 .

# Let's see it graphically

2.

q=100;

# Let's see it graphically

2.

q=100;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | 100 |

# Let's see it graphically

2.

q=100;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | 100 |

3.

p=&q;

# Let's see it graphically

2.

q=100;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | 100 |

3.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | 100 |

p points to q

# Let's see it graphically

2.

q=100;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | 100 |

3.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | 100 |

p points to q

# Guess the output?

```
#include <stdio.h>
int main()
{
   int *p, q;
   p=&q;
   *p=100;
   printf("%d", q);
   return 0;
}
```

# Let's see it graphically

1.

```
int *p, q;
```

Suppose, q is located at memory address 102 and p is right before it, at location 100.

# Let's see it graphically

1.

int *p, q;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | |

Suppose, q is located at memory address 102 and p is right before it, at location 100.

# Let's see it graphically

2.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | |
| 6356744 | unknown |

# Let's see it graphically

2.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | unknown |

p points to q

# Let's see it graphically

2.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | unknown |

p points to q

3.

*p=100;

# Let's see it graphically

2.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | unknown |

p points to q

3.

*p=100;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | |

# Let's see it graphically

2.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | unknown |

p points to q

3.

*p=100;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | |

p points to q

# Let's see it graphically

2.

p=&q;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | unknown |

p points to q

3.

*p=100;

| Location | Contents |
|---|---|
| 6356742 | 6356744 |
| 6356744 | 100 |

p points to q

# Example

```
#include <stdio.h>
int main()
{
   int* pc, c;

   c = 22;
   printf("Address of c: %d\n", &c);
   printf("Value of c: %d\n\n", c);

   pc = &c;
   printf("Address contained in pointer pc: %u\n", pc);
   printf("Value of address contained in pointer pc: %d\n\n", *pc);

   c = 11;
   printf("Address contained in pointer pc: %u\n", pc);
   printf("Value of address contained in pointer pc: %d\n\n", *pc);

   *pc = 2;
   printf("Address of c: %u\n", &c);
   printf("Value of c: %d\n\n", c);
   return 0;
}
```

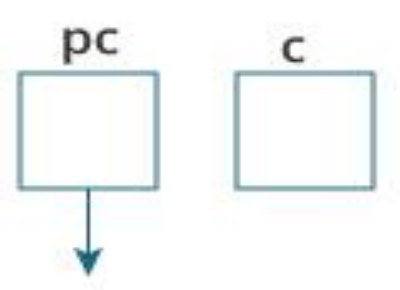
# Example

```
#include <stdio.h>
int main()
{
   int* pc, c;

   c = 22;
   printf("Address of c: %d\n", &c);
   printf("Value of c: %d\n\n", c);

   pc = &c;
   printf("Address contained in pointer pc: %u\n", pc);
   printf("Value of address contained in pointer pc: %d\n\n", *pc);

   c = 11;
   printf("Address contained in pointer pc: %u\n", pc);
   printf("Value of address contained in pointer pc: %d\n\n", *pc);

   *pc = 2;
   printf("Address of c: %u\n", &c);
   printf("Value of c: %d\n\n", c);
   return 0;
}
```

```
Address of c: 6356744
Value of c: 22

Address contained in pointer pc: 6356744
Value of address contained in pointer pc: 22

Address contained in pointer pc: 6356744
Value of address contained in pointer pc: 11

Address of c: 6356744
Value of c: 2
```

# Let's see what actually happens

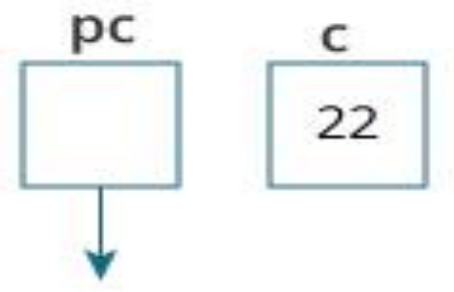
```
int *pc ,c;
```




Here, a pointer pc and a normal variable c, both of type int, is created.
Since pc and c are not initialized at first, pointer pc points to either no address or a random address. And, variable c has an address but contains a random garbage value.

# Let's see what actually happens

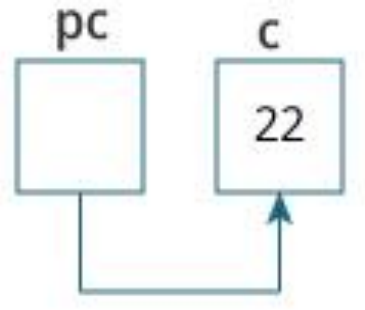
c=22;




22 is stored at the address of variable c.

# Let's see what actually happens

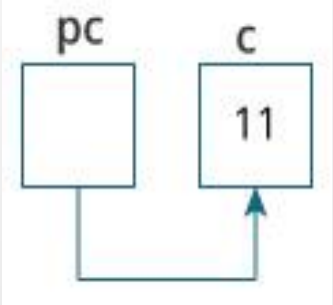
pc=&c;




This assigns the address of variable c to the pointer pc.
Here, the value of pc is same as the address of c.

# Let's see what actually happens

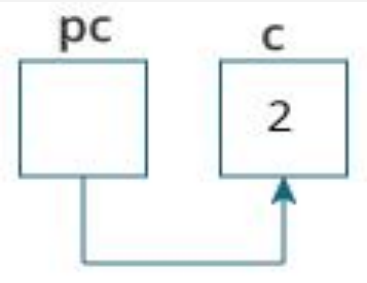
```
c=11;
```




This assigns 11 to variable c.
Since, pointer pc points to the same address as c, value pointed by pointer pc is 11 as well.

# Let's see what actually happens

*pc=2




This change the value at the memory location pointed by pointer pc to 2. Since the address of the pointer pc is same as the address of c,

value of c is also changed to 2.

# Summary

| | | |
|---|---|---|
| int c | c | Value of the variable |
| | &c | Address of the variable |
| int *p | *p | Value of the address contained in the pointer |
| | p | Address that is contained in the pointer |
| | &p | Address of the memory location of the pointer |

# Common mistakes while working with pointer

```
int c, *pc;

// Wrong! pc is address whereas,
// c is not an address.
pc = c;

// Wrong! *pc is the value pointed by address whereas,
// &c is an address.
*pc = &c;

// Correct! pc is an address and,
// &c is also an address.
pc = &c;

// Correct! *pc is the value pointed by address and,
// c is also a value (not address).
*pc = c;
```

# Practice

- Write a program with a for loop that counts from 0 to 9, displaying the number on the screen. Print the numbers using a pointer.