Tutorial 3
Pointer and Array

Jiawei Li (Michael)

Office hours: Tuesday 3:00-5:00pm
Office: PMB426
Email: jiawei.li@nottingham.edu.cn

Pointer

- Define a pointer

```
int *p;
```

- Initialize a pointer

```
p= &a;
```

- Use a pointer

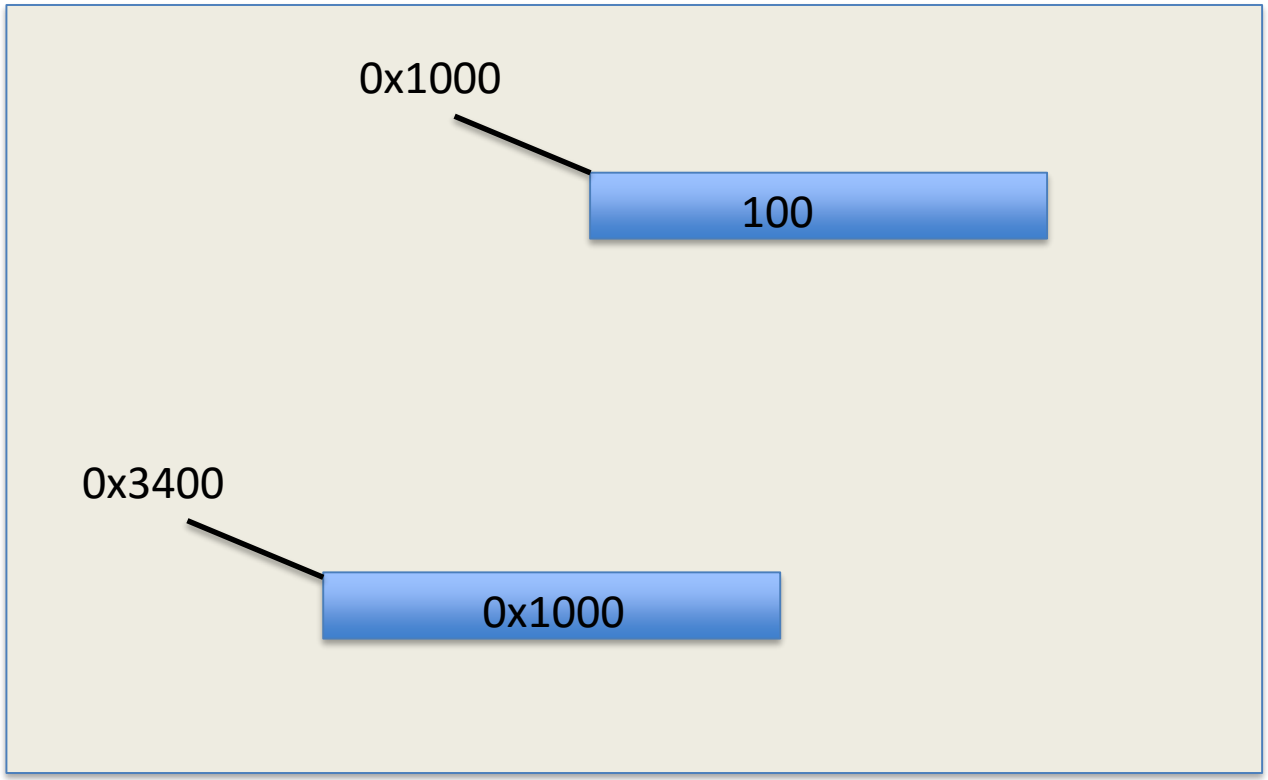
```
*p = 5;   // *p == a
p = &b;  // p -> b
```

Pointer

- A pointer needs to be initialized before using it.
- An initialized pointer is a valid address (not a random value)
- Using an uninitialized pointer frequently leads to ‘segmentation fault’.

Define a variable

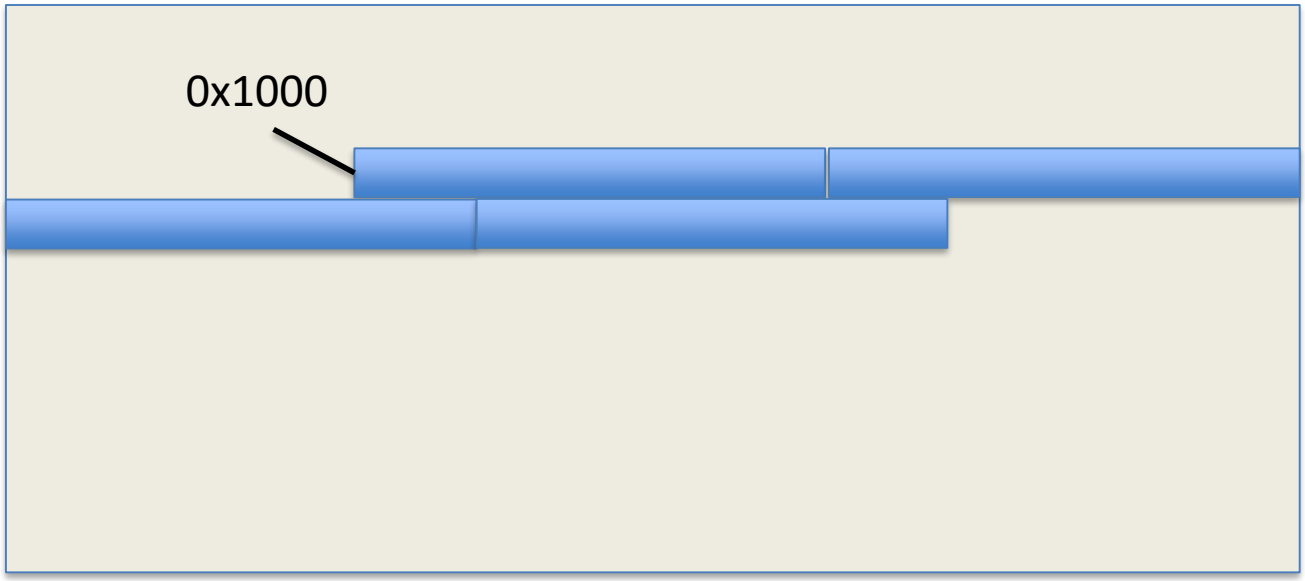
- int a;
- int *p;
- p = &a;
- *p = 100;



Array

An array resembles a constant pointer in many respects. A constant pointer is a constant address that cannot be changed.

```
int a[4];
// a = &b;
// a==0x1000
// a[0] is an int
// &a[1] ==
// 0x1004
```



Frequent mistakes

- Using uninitialized pointers

```
#include<stdio.h>
int main(int argc, char* argv[ ])
{
        int a;
        a = 100;
        printf(“%d”, a);


        int *ptr = NULL;
        *ptr=100;
        printf(“%d, %p”, *ptr, ptr);
}
```

Frequent mistakes

- Pointer to uninitialized variable

```
#include<stdio.h>
int main(int argc, char* argv[ ])
{
        int *ptr;
        int a;
        ptr = &a;

        printf("%d, %d, %p\n", a, *ptr, ptr);

}
```

Exercise 1

```
#include<stdio.h>
int main(int argc, char* argv[ ])
{
	int a = 1;
	int b = 2;

	int *p1 = &a;
	int *p2 = &b;
	printf(“%d, %d”, *p1, *p2);

	p1 = p2;
	*p1 = 3;
	printf(“%d, %d”, a, b);

	printf(“%d, %d”, *p1, *p2);
	return 0;
}
```

Exercise 2

```
#include<stdio.h>
int main(int argc, char* argv[ ])
{
	int a[] = {1, 2, 3};
	printf(“%d, %d, %d\n”, a[0], a[1], a[2]);
	printf(“%p, %p, %p\n”, &a[0], &a[1], &a[2]);

	int b[] = {1, 2, 3};
	printf(“%p, %p\n”, a, b);

	char* s1 = “Paul”;
	char* s2 = “Paul”;
	printf(“%p, %p\n”, s1, s2);


}
```

Exercise for you

- Write a program in C to read a string (no more than 100 characters) from the keyboard, and then count how many spaces are contained, remove all spaces and print out the string in reverse order.