CSE240 – Introduction to Programming Languages (online)

Lecture 07:
Programming with C | Pointers

Javier Gonzalez-Sanchez

javiergs@asu.edu
javiergs.engineering.asu.edu
Office Hours: By appointment

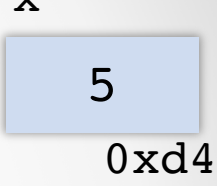
jgs

Pointers

Definitions

- A variable stores a value.
- A pointer is a variable that store an address.
- Direct **manipulation of addresses is powerful** in programming.
- **Pointer type is common in all imperative languages.**
- C has 2 pointer operators: & (ampersand) and * (asterisk)

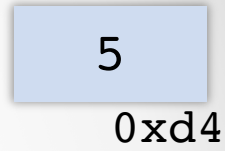
```
int x = 5;
```

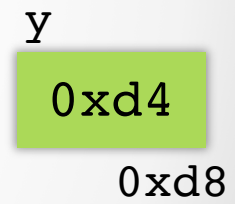


Definitions

- A variable stores a value.
- A pointer is a variable that store an address.
- Direct **manipulation of addresses is powerful** in programming.
- **Pointer type is common in all imperative languages.**
- C has 2 pointer operators: & (ampersand) and * (asterisk)

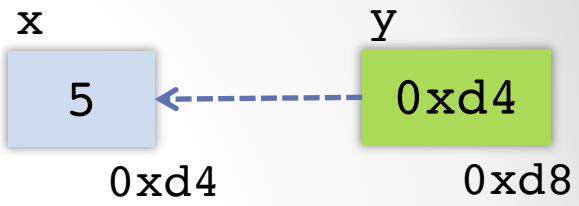
```
int x = 5;

int *y;

y = &x;
```





Example

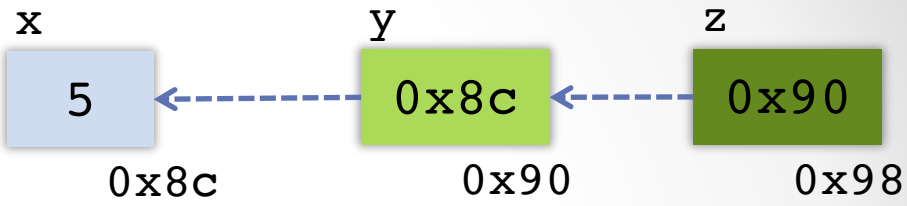
```c
#include <stdio.h>

int main(){
   int x = 5;
   int *y;
   y = &x;
   printf("value of x: %d \n", x);
   printf("address of x: %p \n", &x);
   printf("value of y: %p \n", y);
   printf("address of y: %p \n", &y);
   printf("value pointed by y: %d \n", *y);
   return 0;
}
```




```
value of x: 5
address of x: 0x7ffc1dcf44d4
value of y: 0x7ffc1dcf44d4
address of y: 0x7ffc1dcf44d8
value pointed by y: 5
```

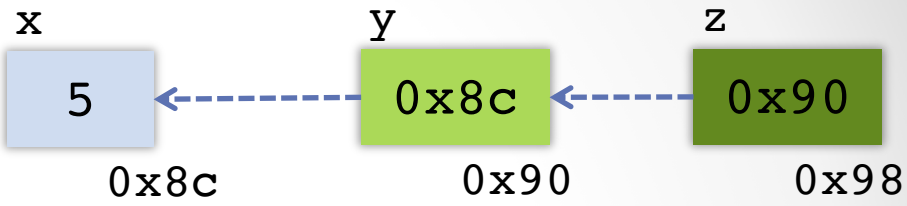
Example


```
#include <stdio.h>

int main(){
   int x = 5;
   int *y = &x;
   int **z = &y;
   printf("value of x: %d \n", x);
   printf("address of x: %p \n", &x);
   printf("value of y: %p \n", y);
   printf("address of y: %p \n", &y);
   printf("value pointed by y: %d \n", *y);
   printf("value of z: %p \n", z);
   printf("address of z: %p \n", &z);
   printf("value pointed by z: %p \n", *z);
   printf("value pointed by the address pointed by z: %d \n", **z);
   return 0;
}
```



Example

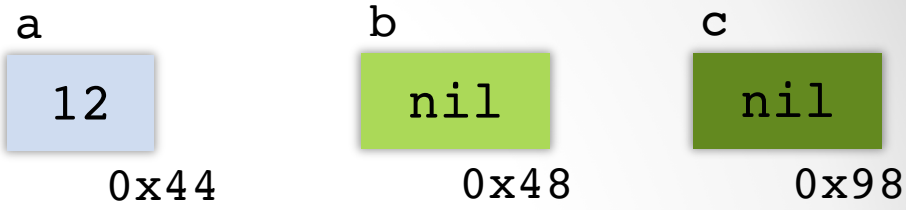

```c
#include <stdio.h>

int main(){
   int x = 5;
   int *y = &x;
   int **z = &y;
   printf("value of x: %d \n", x);
   printf("address of x: %p \n", &x);
   printf("value of y: %p \n", y);
   printf("address of y: %p \n", &
   printf("value pointed by y: %d
   printf("value of z: %p \n", z);
   printf("address of z: %p \n", &
   printf("value pointed by z: %p
   printf("value pointed by the ad
   return 0;
}
```




```
value of x: 5
address of x: 0x7ffeec7c618c
value of y: 0x7ffeec7c618c
address of y: 0x7ffeec7c6190
value pointed by y: 5
value of z: 0x7ffeec7c6190
address of z: 0x7ffeec7c6198
value pointed by z: 0x7ffeec7c618c
value pointed by the addess pointed by z: 5
```

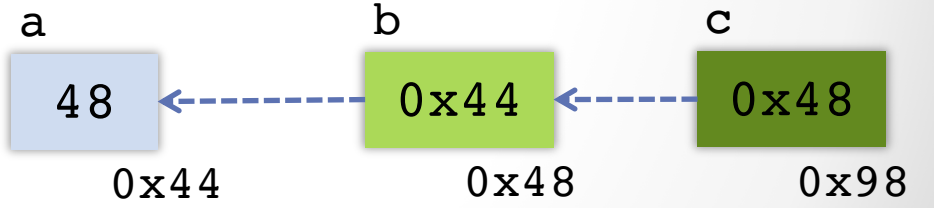
Example

```
#include <stdio.h>

int main() {

  int a = 12, *b = 0, **c = 0;
  printf("a = %d, b = %p, c = %p\n", a, b, c);
  b = &a;  *b = 24;
  c= &b;  **c = 48;
  printf("a = %d, b = %p, c = %p\n", a, b, c);
  return 0;

}
```



Example

```
#include <stdio.h>

int main() {

   int a = 12, *b = 0, **c = 0;
   printf("a = %d, b = %p, c = %p\n", a, b, c);
   b = &a;  *b = 24;
   c= &b;  **c = 48;
   printf("a = %d, b = %p, c = %p\n", a, b, c);
   return 0;

}
```




```
a = 12, b = (nil), c = (nil)
a = 48, b = 0x7ffe6ca52244, c = 0x7ffe6ca52248
```

Pointers

- & is a *referencing function* that returns the address value of the variable it precedes, For instance:

  if integer x is allocated at memory address = 2000, then
  y = &x and y = 2000.

- * represents the variable name for a given address.

  y = &x; y = 100
  *y is an alias of x.
  *y = 0 and x = 0.
  &(*p) is the same that p

Arrays are Pointers

jgs

- An array is a pointer to a set of consecutive elements

`a[0]` is the same that `*(a+0)`
`a[1]` is the same that `*(a+1)`
`a[2]` is the same that `*(a+2)`
`a[3]` is the same that `*(a+3)`
`etc.`

```
int a [6];
```

Arrays are Pointers

jgs

```
#include <stdio.h>

void main() {
  int i = 0;
  char a[ ] = "Hello CSE 240";
  printf("\n message: %s\n ", a);
  while (a[i] != '\0') { a[i] = *(a + i)+1; i++;}
  printf("\n message after encryption: %s\n  ", a);
  char *q;
  q = a;
  while (*q != '\0') { *q = *q-1; q++;}
  printf("\n message after decryption:  %s\n ", a);
}
```

Arrays are Pointers

jgs

```
#include <stdio.h>

void main() {
  int i = 0;
  char a[ ] = "Hello CSE 240";
  printf("\n message: %s\n ", a);
  while (a[i] != '\0') { a[i] = *(a + i)+1; i++;}
  printf("\n message after encryption: %s\n  ", a);
  char *q;
  q = a;
  while (*q != '\0') { *q = *q-1; q++;}
  printf("\n message after decryption:  %s\n ", a);
}
```

```
a[i] = *(a + i)+ 1

a[i] = a[i]+ 1

*(a+i) = a[i]+ 1

*(a+i) = *(a+i)+ 1
```

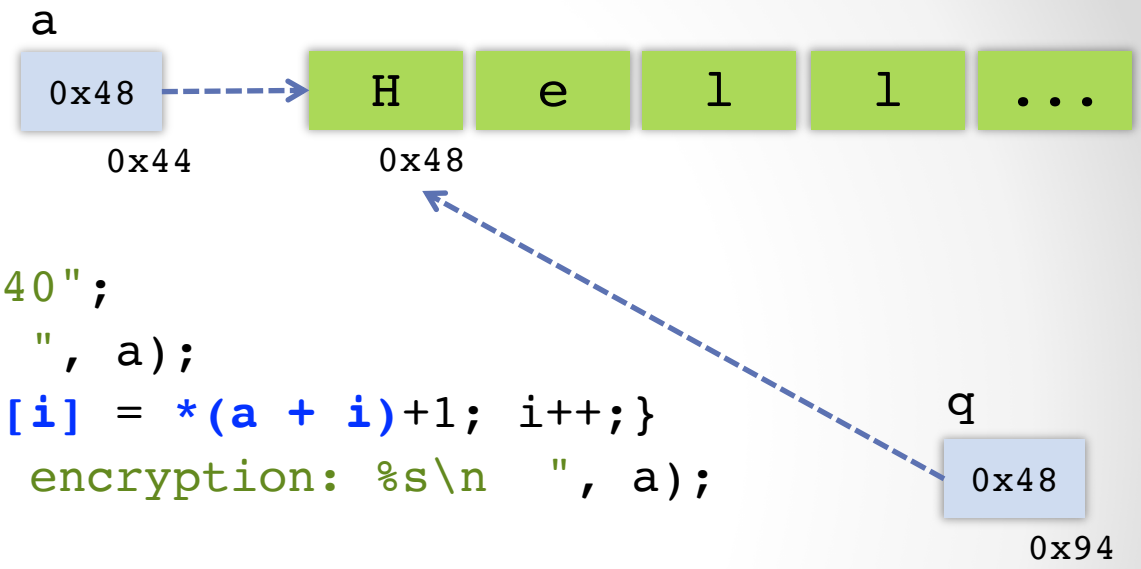
Arrays are Pointers

jgs

```
#include <stdio.h>

void main() {
  int i = 0;
  char a[ ] = "Hello CSE 240";
  printf("\n message: %s\n ", a);
  while (a[i] != '\0') { a[i] = *(a + i)+1; i++;}
  printf("\n message after encryption: %s\n  ", a);
  char *q;
  q = a;
  while (*q != '\0') { *q = *q-1; q++;}
  printf("\n message after decryption:  %s\n ", a);
}
```

Arrays are Pointers

```
#include <stdio.h>

void main() {
  int i = 0;
  char a[ ] = "Hello CSE 240";
  printf("\n message: %s\n ", a);
  while (a[i] != '\0') { a[i] = *(a + i)+1; i++;}
  printf("\n message after encryption: %s\n  ", a);
  char *q;
  q = a;
  while (*q != '\0') { *q = *q-1; q++;}
  printf("\n message after decryption:  %s\n ", a);
}
```



Arrays are Pointers

jgs

```
#include <stdio.h>

void main() {
  int i = 0;
  char a[ ] = "Hello CSE 240";
  printf("\n message: %s\n ", a);
  while (a[i] != '\0') { a[i] = *(a + i)+1; i++;}
  printf("\n message after encryption: %s\n  ", a);
  char *q;
  q = a;
  while (*q != '\0') { *q = *q-1
  printf("\n message after decry
}
```

```
message: Hello CSE 240

message after encryption: Ifmmp!DTF!351

message after decryption:  Hello CSE 240
```

CSE240 – Introduction to Programming Languages (online)

Javier Gonzalez-Sanchez
javiergs@asu.edu
Fall 2017