# ESC 101: FUNDAMENTALS OF COMPUTING

Lecture 21

Feb 18, 2010

LECTURE 21 () ESc 101 FEB 18, 2010 1 / 11

# **OUTLINE**

POINTERS

2 SCANF

Lecture 21 () ESc 101 Feb 18, 2010 2 / 11

## Pointer Arithmetic

- Since pointer variables store memory addresses, we can add and subtract from them to access other addresses!
- ullet For a pointer variable y, \*(y+1) refers to the next memory location.
- Depending on the type of variable, this can be one or more bytes away.
- Caution: This must be done with extreme care!!
- If we shift the pointer to a location outside the designated memory locations then unpredictable things may happen.

LECTURE 21 () ESc 101 FEB 18, 2010 3 / 11

## Pointer Arithmetic

- Since pointer variables store memory addresses, we can add and subtract from them to access other addresses!
- For a pointer variable y, \*(y+1) refers to the next memory location.
- Depending on the type of variable, this can be one or more bytes away.
- Caution: This must be done with extreme care!!
- If we shift the pointer to a location outside the designated memory locations then unpredictable things may happen.

## POINTER ARITHMETIC

- Since pointer variables store memory addresses, we can add and subtract from them to access other addresses!
- For a pointer variable y, \*(y+1) refers to the next memory location.
- Depending on the type of variable, this can be one or more bytes away.
- Caution: This must be done with extreme care!!
- If we shift the pointer to a location outside the designated memory locations then unpredictable things may happen.

## Pointer Arithmetic

- Since pointer variables store memory addresses, we can add and subtract from them to access other addresses!
- For a pointer variable y, \*(y+1) refers to the next memory location.
- Depending on the type of variable, this can be one or more bytes away.
- Caution: This must be done with extreme care!!
- If we shift the pointer to a location outside the designated memory locations then unpredictable things may happen.

```
int main()
1343730016
                                 {
1343730012
1343730008
                                      int n = 10;
1343730004
                                      int m = 5;
1343730000
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
                                      /* Do something */
```

# **MEMORY**

```
int main()
1343730016
1343730012
                  10
1343730008
                                      int n = 10;
1343730004
                                      int m = 5;
1343730000
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
                                      /* Do something */
```

## **MEMORY**

ECTURE 21 () ESc 101 Feb 18, 2010 4 / 11

```
int main()
1343730016
1343730012
                  10
1343730008
                                     int n = 10;
1343730004
                                     int m = 5;
1343730000
                                     int z[2];
1343729996
1343729992
1343729988
                                     for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
                                     /* Do something */
```

# **MEMORY**

Lecture 21 () ESc 101 Feb 18, 2010 4 / 11

```
int main()
1343730016
1343730012
                  10
1343730008
                                      int n = 10;
1343730004
                                      int m = 5;
1343730000
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                      /* Do something */
```

# **MEMORY**

◆□ → ←同 → ← 三 → ○○○

```
int main()
1343730016
1343730012
                  10
1343730008
                                      int n = 10;
1343730004
                                      int m = 5;
1343730000
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                      /* Do something */
                                 i = 0, setting z[0] or *z
              MEMORY
```

(ロ) (部) (注) (注) 注 の(で)

```
int main()
1343730016
1343730012
                  10
1343730008
                                      int n = 10;
1343730004
                                      int m = 5;
1343730000
                  100
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                      /* Do something */
                                 i = 0, setting z[0] or *z
              MEMORY
```

4日 → 4団 → 4 三 → 4 三 → 9 Q ○

LECTURE 21 () ESc 101 Feb 18, 2010 4 / 11

```
int main()
1343730016
1343730012
                  10
1343730008
                                     int n = 10;
1343730004
                                     int m = 5;
1343730000
                 100
                                     int z[2];
1343729996
1343729992
1343729988
                                     for (int i=0; i<4; i++)
1343729984
                                         z[i] = 100;
1343729980
              1343730000
                                     /* Do something */
                                 i = 1, setting z[1] or
              MEMORY
```

```
int main()
1343730016
1343730012
                  10
1343730008
                                     int n = 10;
1343730004
                 100
                                     int m = 5;
1343730000
                 100
                                     int z[2];
1343729996
1343729992
1343729988
                                     for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                     /* Do something */
                                 i = 1, setting z[1] or
              MEMORY
                                 *(z+1)
```

←□ → ←□ → ← □ → ← □ → ← □

```
int main()
1343730016
1343730012
                  10
1343730008
                                     int n = 10;
1343730004
                 100
                                     int m = 5;
1343730000
                 100
                                     int z[2];
1343729996
1343729992
1343729988
                                     for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                     /* Do something */
                                 i = 2, setting z[2] or
              MEMORY
```

21 () ESc 101 FEB 18, 2010 4 / 11

```
int main()
1343730016
1343730012
                  10
1343730008
                 100
                                      int n = 10;
1343730004
                 100
                                      int m = 5;
1343730000
                 100
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                      /* Do something */
                                 i = 2, setting z[2] or
              MEMORY
                                 *(z+2)
```

4 D > 4 B > 4 E >

```
int main()
1343730016
1343730012
                  10
1343730008
                 100
                                     int n = 10;
1343730004
                 100
                                     int m = 5;
1343730000
                 100
                                     int z[2];
1343729996
1343729992
1343729988
                                     for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                     /* Do something */
                                 i = 3, setting z[3] or
              MEMORY
```

LECTURE 21 () ESc 101 Feb 18, 2010 4 / 11

```
int main()
1343730016
                  100
1343730012
1343730008
                  100
                                      int n = 10;
1343730004
                  100
                                      int m = 5;
1343730000
                 100
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                      /* Do something */
                                 i = 3, setting z[3] or
              MEMORY
                                 *(z+3)
```

```
int main()
1343730016
1343730012
                 100
1343730008
                 100
                                     int n = 10;
1343730004
                 100
                                     int m = 5;
1343730000
                 100
                                     int z[2];
1343729996
1343729992
1343729988
                                     for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                     /* Do something */
                                 i = 4
              MEMORY
```

4□ > 4□ > 4 = > 4 = > = 90

ECTURE 21 () ESc 101 Feb 18, 2010 4 / 11

```
int main()
1343730016
1343730012
                  100
1343730008
                  100
                                      int n = 10;
1343730004
                  100
                                      int m = 5;
1343730000
                  100
                                      int z[2];
1343729996
1343729992
1343729988
                                      for (int i=0; i<4; i++)
1343729984
                                          z[i] = 100;
1343729980
              1343730000
                                      /* Do something */
```

**MEMORY** 

Lecture 21 () ESc 101 Feb 18, 2010 4 / 11

# **OUTLINE**

1 Pointers

2 SCANF

#### scanf FUNCTION

The general format of scanf is:

```
scanftf( <string constant>, argument-1, ..., argument-k )
```

Lecture 21 () ESc 101 Feb 18, 2010 6 / 11

- The <string constant> is a constant string specifying what needs to be read as input.
- It contains special commands, each starting with %.
- There are exactly k special commands.



Lecture 21 () ESc 101 Feb 18, 2010 7 / 11

- The <string constant> is a constant string specifying what needs to be read as input.
- It contains special commands, each starting with %.
- There are exactly k special commands.



LECTURE 21 () ESc 101 Feb 18, 2010 7 / 11

## Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

#### Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

◆ロ → ← 同 → ← 三 → へ 回 → の へ ○

LECTURE 21 () ESc 101 FEB 18, 2010 8 / 11

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

◆ロ → ◆問 → ◆ き → ◆ き ・ か Q (~)

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

◆ロ → ← 同 → ← 三 → へ 回 → の へ ○

Lecture 21 () ESc 101 Feb 18, 2010 8 / 11

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

◆□ → ◆問 → ◆ き → を ● め へ ○

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

↓□▶ ↓□▶ ↓□▶ ↓□▶ □ ♥Q○

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

Suppose <string constant> is:

"<s1>%d<s2>%c<s3>%s<s4>%f<s5>"

## Its meaning is:

- Read string <s1>,
- Read an integer and store it in \*argument-1,
- Read string <s2>,
- Read a symbol and store it in \*argument-2,
- Read string <s3>,
- Read a string and store it in the array argument-3,
- Read string <s4>,
- Read a real number and store it in \*argument-4, ...

- (ロ) (部) (注) (注) 注 り(()

Lecture 21 () ESc 101 Feb 18, 2010 8 / 11

```
For the call:
scanf("Number = %d, string = %s", &n, str);
the input must be one of the following:
Number = 35, string = test
Number
                     35, string
                                              test
Number=35, string=test
The following inputs are wrong:
number = 35, string = test
Number = 35, string test
```

```
For the call:

scanf("Number = %d, string = %s\n", &n, str);

the input must be of the following form:
```

This is because the  $\n$  at the end of format string matches with any number of whitespaces.

Lecture 21 () ESc 101 Feb 18, 2010 10 / 11

#### Consider the program:

```
main()
{
    char str[10];
    scanf("%s", str);
    printf("%s", str);
}
```

- Suppose its input is: abcdefghijklmnop, which is a 16 character long string.
- This may well be accepted by scanf and stored in the array str!
- The reason is the same as before: if the extra memory locations are inside the program, then those locations will be overwritten.

LECTURE 21 () ESC 101 FEB 18, 2010 11 / 11

Consider the program:

```
main()
{
    char str[10];
    scanf("%s", str);
    printf("%s", str);
}
```

- Suppose its input is: abcdefghijklmnop, which is a 16 character long string.
- This may well be accepted by scanf and stored in the array str!
- The reason is the same as before: if the extra memory locations are inside the program, then those locations will be overwritten.

LECTURE 21 () ESc 101 Feb 18, 2010 11 / 11

Consider the program:

```
main()
{
    char str[10];
    scanf("%s", str);
    printf("%s", str);
}
```

- Suppose its input is: abcdefghijklmnop, which is a 16 character long string.
- This may well be accepted by scanf and stored in the array str!
- The reason is the same as before: if the extra memory locations are inside the program, then those locations will be overwritten.

Consider the program:

```
main()
{
    char str[10];
    scanf("%s", str);
    printf("%s", str);
}
```

- Suppose its input is: abcdefghijklmnop, which is a 16 character long string.
- This may well be accepted by scanf and stored in the array str!
- The reason is the same as before: if the extra memory locations are inside the program, then those locations will be overwritten.