Appendix: Other operations

Copyright ©: University of Illinois CS 241 Staff

Increment & decrement

- `x++`: yield old value, add one
- `++x`: add one, yield new value

```
int x = 10;
x++;
int y = x++;

int z = ++x;
```

11

13

- `--x` and `x--` are similar (subtract one)

Copyright ©: University of Illinois CS 241 Staff

Math: Increment and Decrement Operators

- Example 1:

```
int x, y, z, w;
y=10; w=2;
x=++y;
z=--w;
```

- Example 2:

```
int x, y, z, w;
y=10; w=2;
x=y++;
z=w--;
```

What are **x** and **y** at the end of each example?

Math: Increment and Decrement Operators

- Example 1:

```
int x, y, z, w;
y=10; w=2;
x=++y;
z=--w;
```

- First increment/ decrement, then assign result
- `x` is 11, `z` is 1

- Example 2:

```
int x, y, z, w;
y=10; w=2;
x=y++;
z=w--;
```

- First assign result, then increment/ decrement
- `x` is 10, `z` is 2

Copyright ©: University of Illinois CS 241 Staff

Math: Increment and Decrement Operators on Pointers

- Example

```
int a[2];
int number1, number2, *p;
a[0]=1; a[1]=10;
p=a;
number1 = *p++;
number2 = *p;
```

- What will `number1` and `number2` be at the end?

Copyright ©: University of Illinois CS 241 Staff

Math: Increment and Decrement Operators on Pointers

- Example

```
int a[2];
int number1, number2, *p;
a[0]=1; a[1]=10;
p=a;
number1 = *p++;
number2 = *p;
```

Hint: ++ increments pointer p not variable *p (pointing to the line `number1 = *p++;`)

- What will `number1` and `number2` be at the end?

Copyright ©: University of Illinois CS 241 Staff

Logic: Relational (Condition) Operators

`==`	equal to
`!=`	not equal to
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to

Copyright ©: University of Illinois CS 241 Staff

Logic Example

```
if (a == b)
   printf ("Equal.");
else
   printf ("Not Equal.");
```

- Question: what will happen if I replaced the above with:

```
if (a = b)
   printf ("Equal.");
else
   printf ("Not Equal.");
```

Copyright ©: University of Illinois CS 241 Staff

Logic Example

```
if (a == b)
   printf (“Equal.”);
else
   printf (“Not Equal.”);
```

- Question: what will happen if I replaced the above with:

```
if (a = b)
   printf (“Equal.”);
else
   printf (“Not Equal.”);
```

Perfectly LEGAL C statement! (syntactically speaking)
It copies the value of `b` into `a`. The statement will be interpreted as `TRUE` if `b` is non-zero.

Copyright ©: University of Illinois CS 241 Staff

strcpy, strlen

- `strcpy(ptr1, ptr2);`
  - `ptr1` and `ptr2` are pointers to char
- `value = strlen(ptr);`
  - `value` is an integer
  - `ptr` is a pointer to char

```
int len;
char str[15];
strcpy (str, "Hello, world!");
len = strlen(str);
```

Copyright ©: University of Illinois CS 241 Staff

strncpy

- `strncpy(ptr1, ptr2, num);`
  - `ptr1` and `ptr2` are pointers to char
  - `num` is the number of characters to be copied

```
int len;
char str1[15],
   str2[15];
strcpy (str1,
   "Hello, world!");
strncpy (str2, str1,
   5);
```

strncpy

- `strncpy(ptr1, ptr2, num);`
  - `ptr1` and `ptr2` are pointers to char
  - `num` is the number of characters to be copied

```
int len;
char str1[15],
   str2[15];
strcpy (str1,
   "Hello, world!");
strncpy (str2, str1,
   5);
```

Caution: strncpy blindly copies the characters. It does not voluntarily append the string-terminating null character.

Copyright ©: University of Illinois CS 241 Staff

`strcat`

- `strcat(ptr1, ptr2);`
  - `ptr1` and `ptr2` are pointers to char

- Concatenates the two null terminated strings yielding one string (pointed to by `ptr1`).

```
char S[25] = "world!";
char D[25] = "Hello, ";
strcat(D, S);
```

Copyright ©: University of Illinois CS 241 Staff

`strcat`

- `strcat(ptr1, ptr2);`
  - `ptr1` and `ptr2` are pointers to char
- Concatenates the two null terminated strings yielding one string (pointed to by `ptr1`).
  - Find the end of the destination string
  - Append the source string to the end of the destination string
  - Add a NULL to new destination string

Copyright ©: University of Illinois CS 241 Staff

strcat Example

- What's wrong with

```
char S[25] = "world!";
strcat("Hello, ", S);
```

strcat Example

- What' s wrong with

```
char *s = malloc(11 * sizeof(char));
        /* Allocate enough memory for an
           array of 11 characters, enough
           to store a 10-char long string. */
strcat(s, "Hello");
strcat(s, "World");
```

Copyright ©: University of Illinois CS 241 Staff

strcat Example

- What’s wrong with

```
char *s = malloc(11 * sizeof(char));
        /* Allocate enough memory for an
           array of 11 characters, enough
           to store a 10-char long string. */
s[0] = 0;
strcat(s, "Hello"); or strcpy(s, "Hello");
strcat(s, "World");
```

Copyright ©: University of Illinois CS 241 Staff

strcat

- `strcat(ptr1, ptr2);`
  - `ptr1` and `ptr2` are pointers to char
- Compare to Java and C++
  - `string s = s + " World!";`
- What would you get in C?
  - If you did `char* ptr0 = ptr1+ptr2;`

strcat

- `strcat(ptr1, ptr2);`
  - `ptr1` and `ptr2` are pointers to char

- Compare to Java and C++
  - `string s = s + " World!";`

- What would you get in C?
  - If you did `char* ptr0 = ptr1+ptr2;`
  - You would get the sum of two memory locations!

`strcmp`

- `diff = strcmp(ptr1, ptr2);`
  - `diff` is an integer
  - `ptr1` and `ptr2` are pointers to char
- Returns
  - zero if strings are identical
  - < 0 if `ptr1` is less than `ptr2` (earlier in a dictionary)
  - > 0 if `ptr1` is greater than `ptr2` (later in a dictionary)

```
int diff;
char s1[25] = "pat";
char s2[25] = "pet";
diff = strcmp(s1, s2);
```

Copyright ©: University of Illinois CS 241 Staff

<table>
<tr><th>Operator</th><th>Description</th><th>Associativity</th></tr>
<tr><td>()
[]
.
->
++ --</td><td>Parentheses (function call)
Brackets (array subscript)
Member selection via object name
Member selection via pointer
Postfix increment/decrement</td><td>left-to-right</td></tr>
<tr><td>++ --
+ -
! ~
(type)
*
&
sizeof</td><td>Prefix increment/decrement
Unary plus/minus
Logical negation/bitwise complement
Cast (change type)
Dereference
Address
Determine size in bytes</td><td>right-to-left</td></tr>
<tr><td>* / %</td><td>Multiplication/division/modulus</td><td>left-to-right</td></tr>
<tr><td>+ -</td><td>Addition/subtraction</td><td>left-to-right</td></tr>
<tr><td><< >></td><td>Bitwise shift left, Bitwise shift right</td><td>left-to-right</td></tr>
<tr><td>< <=
> >=</td><td>Relational less than/less than or equal to
Relational greater than/greater than or equal to</td><td>left-to-right</td></tr>
<tr><td>== !=</td><td>Relational is equal to/is not equal to</td><td>left-to-right</td></tr>
<tr><td>&</td><td>Bitwise AND</td><td>left-to-right</td></tr>
<tr><td>^</td><td>Bitwise exclusive OR</td><td>left-to-right</td></tr>
<tr><td>|</td><td>Bitwise inclusive OR</td><td>left-to-right</td></tr>
<tr><td>&&</td><td>Logical AND</td><td>left-to-right</td></tr>
<tr><td>||</td><td>Logical OR</td><td>left-to-right</td></tr>
<tr><td>?:</td><td>Ternary conditional</td><td>right-to-left</td></tr>
<tr><td>=
+= -=
*= /=
%= &=
^= |=
<<= >>=</td><td>Assignment
Addition/subtraction assignment
Multiplication/division assignment
Modulus/bitwise AND assignment
Bitwise exclusive/inclusive OR assignment
Bitwise shift left/right assignment</td><td>right-to-left</td></tr>
<tr><td>,</td><td>Comma (separate expressions)</td><td>left-to-right</td></tr>
</table>

Questions

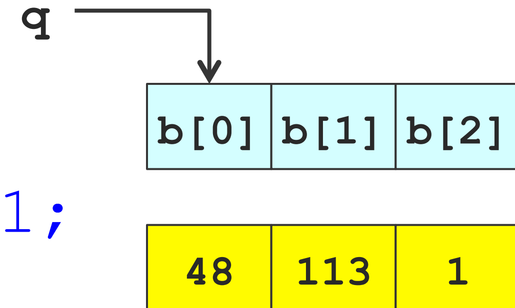
- What is the value of `b[2]` at the end?

```
int b[3];
int* q;

b[0]=48; b[1]=113; b[2]=1;

q=b;

*(q+1)=2;

b[2]=*b;

b[2]=b[2]+b[1];
```

Questions

- What is the value of `b[2]` at the end?

```
int b[3];
int* q;

b[0]=48; b[1]=113; b[2]=1;

q=b;

*(q+1)=2;

b[2]=*b;

b[2]=b[2]+b[1];
```

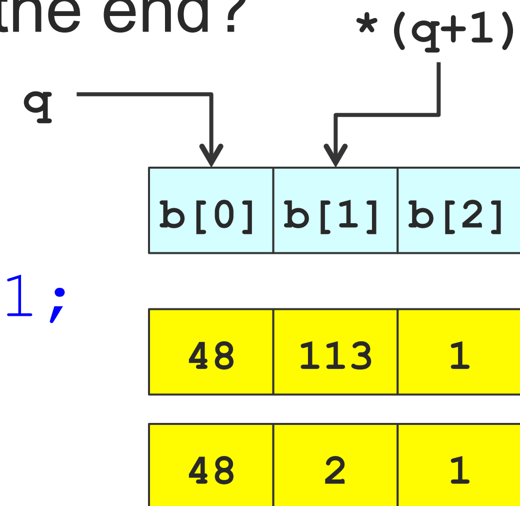


Copyright ©: University of Illinois CS 241 Staff

Questions

- What is the value of `b[2]` at the end?

```
int b[3];
int* q;

b[0]=48; b[1]=113; b[2]=1;

q=b;

*(q+1)=2;

b[2]=*b;

b[2]=b[2]+b[1];
```

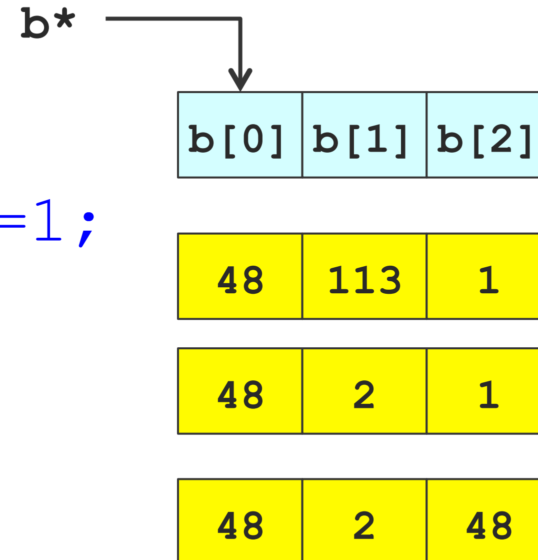


Copyright ©: University of Illinois CS 241 Staff

Questions

- What is the value of `b[2]` at the end?

```
int b[3];
int* q;

b[0]=48; b[1]=113; b[2]=1;

q=b;

*(q+1)=2;

b[2]=*b;

b[2]=b[2]+b[1];
```

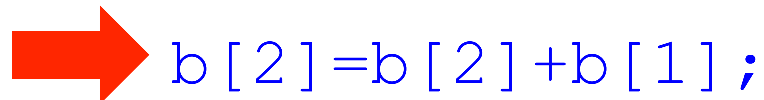


Copyright ©: University of Illinois CS 241 Staff

Questions

- What is the value of `b[2]` at the end?

```
int b[3];
int* q;

b[0]=48; b[1]=113; b[2]=1;

q=b;

*(q+1)=2;

b[2]=*b;
```



```
b[2]=b[2]+b[1];
```

b[0]	b[1]	b[2]
48	113	1
48	2	1
48	2	48
48	2	50

Copyright ©: University of Illinois CS 241 Staff