```
 1: /*-------------------------------------------------------------------------*/
 2: /* rev.c                                                                   */
 3: /* Author: Bob Dondero                                                     */
 4: /*-------------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7:
 8: /*-------------------------------------------------------------------------*/
 9:
10: /* Read ARRAY_LENGTH integers from stdin, and write them in reverse
11:    order to stdout.  Return 0. */
12:
13: int main(void)
14: {
15:    enum {ARRAY_LENGTH = 5};
16:
17:    int aiNums[ARRAY_LENGTH];
18:    int i;
19:
20:    printf("Enter %d integers:\n", ARRAY_LENGTH);
21:    for (i = 0; i < ARRAY_LENGTH; i++)
22:       scanf("%d", &aiNums[i]);
23:
24:    printf("\n");
25:
26:    printf("The integers in reverse order are:\n");
27:    for (i = ARRAY_LENGTH-1; i >= 0; i--)
28:       printf("%d\n", aiNums[i]);
29:
30:    return 0;
31: }
```

revp1.c (Page 1 of 1)

```
 1: /*-------------------------------------------------------------------------*/
 2: /* revp1.c                                                                 */
 3: /* Author: Bob Dondero                                                     */
 4: /*-------------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7:
 8: /*-------------------------------------------------------------------------*/
 9:
10: /* Read ARRAY_LENGTH integers from stdin, and write them in reverse
11:    order to stdout.  Return 0. */
12:
13: int main(void)
14: {
15:    enum {ARRAY_LENGTH = 5};
16:
17:    int aiNums[ARRAY_LENGTH];
18:    int i;
19:
20:    printf("Enter %d integers:\n", ARRAY_LENGTH);
21:    for (i = 0; i < ARRAY_LENGTH; i++)
22:       scanf("%d", aiNums + i);
23:
24:    printf("\n");
25:
26:    printf("The integers in reverse order are:\n");
27:    for (i = ARRAY_LENGTH-1; i >= 0; i--)
28:       printf("%d\n", *(aiNums + i));
29:
30:    return 0;
31: }
```

revp2.c (Page 1 of 1)

```
 1: /*-------------------------------------------------------------------------*/
 2: /* revp2.c                                                                 */
 3: /* Author: Bob Dondero                                                     */
 4: /*-------------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7:
 8: /*-------------------------------------------------------------------------*/
 9:
10: /* Read ARRAY_LENGTH integers from stdin, and write them in reverse
11:    order to stdout.  Return 0. */
12:
13: int main(void)
14: {
15:    enum {ARRAY_LENGTH = 5};
16:
17:    int aiNums[ARRAY_LENGTH];
18:    int *piCurrent;
19:    int *piAfterLast;
20:
21:    piAfterLast = aiNums + ARRAY_LENGTH;
22:
23:    printf("Enter %d integers:\n", ARRAY_LENGTH);
24:    piCurrent = aiNums;
25:    while (piCurrent != piAfterLast)  /* Could use "for" statement. */
26:    {
27:       scanf("%d", piCurrent);
28:       piCurrent++;
29:    }
30:
31:    printf("\n");
32:
33:    printf("The integers in reverse order are:\n");
34:    while (piCurrent != aiNums)
35:    {
36:       piCurrent--;
37:       printf("%d\n", *piCurrent);
38:    }
39:
40:    return 0;
41: }
```

revfn1.c (Page 1 of 1)

```
 1: /*--------------------------------------------------------------------*/
 2: /* revfn1.c                                                           */
 3: /* Author: Bob Dondero                                                */
 4: /*--------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7: #include <assert.h>
 8:
 9: /*--------------------------------------------------------------------*/
10:
11: /* The number of elements in the array. */
12: enum {ARRAY_LENGTH = 5};
13:
14: /*--------------------------------------------------------------------*/
15:
16: /* piNums is a pointer to the 0th element of an array.  Read
17:    integers from stdin into that array.  */
18:
19: static void getNumbers(int *piNums)
20: {
21:    int i;
22:
23:    assert(piNums != NULL);
24:
25:    for (i = 0; i < ARRAY_LENGTH; i++)
26:       scanf("%d", &piNums[i]);
27: }
28:
29: /*--------------------------------------------------------------------*/
30:
31: /* piNums is a pointer to the 0th element of an array.  Print the
32:    elements of that array to stdout in reverse order.  */
33:
34: static void putNumbers(int *piNums)
35: {
36:    int i;
37:
38:    assert(piNums != NULL);
39:
40:    for (i = ARRAY_LENGTH-1; i >= 0; i--)
41:       printf("%d\n", piNums[i]);
42: }
43:
44: /*--------------------------------------------------------------------*/
45:
46: /* Read ARRAY_LENGTH integers from stdin, and write them in reverse
47:    order to stdout.  Return 0. */
48:
49: int main(void)
50: {
51:    int aiNums[ARRAY_LENGTH];
52:
53:    printf("Enter %d integers:\n", ARRAY_LENGTH);
54:    getNumbers(aiNums);
55:
56:    printf("\n");
57:    printf("The integers in reverse order are:\n");
58:    putNumbers(aiNums);
59:
60:    return 0;
61: }
```

revfn2.c (Page 1 of 1)

```
 1: /*--------------------------------------------------------------------*/
 2: /* revfn2.c                                                           */
 3: /* Author: Bob Dondero                                                */
 4: /*--------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7: #include <assert.h>
 8:
 9: /*--------------------------------------------------------------------*/
10:
11: /* The number of elements in the array. */
12: enum {ARRAY_LENGTH = 5};
13:
14: /*--------------------------------------------------------------------*/
15:
16: /* piNums is a pointer to the 0th element of an array.  Read
17:    integers from stdin into that array.  */
18:
19: static void getNumbers(int piNums[ARRAY_LENGTH])
20: {
21:    int i;
22:
23:    assert(piNums != NULL);
24:
25:    for (i = 0; i < ARRAY_LENGTH; i++)
26:       scanf("%d", &piNums[i]);
27: }
28:
29: /*--------------------------------------------------------------------*/
30:
31: /* piNums is a pointer to the 0th element of an array.  Print the
32:    elements of that array to stdout in reverse order.  */
33:
34: static void putNumbers(int piNums[ARRAY_LENGTH])
35: {
36:    int i;
37:
38:    assert(piNums != NULL);
39:
40:    for (i = ARRAY_LENGTH-1; i >= 0; i--)
41:       printf("%d\n", piNums[i]);
42: }
43:
44: /*--------------------------------------------------------------------*/
45:
46: /* Read ARRAY_LENGTH integers from stdin, and write them in reverse
47:    order to stdout.  Return 0. */
48:
49: int main(void)
50: {
51:    int aiNums[ARRAY_LENGTH];
52:
53:    printf("Enter %d integers:\n", ARRAY_LENGTH);
54:    getNumbers(aiNums);
55:
56:    printf("\n");
57:    printf("The integers in reverse order are:\n");
58:    putNumbers(aiNums);
59:
60:    return 0;
61: }
```

revfn3.c (Page 1 of 1)

```
 1: /*--------------------------------------------------------------------*/
 2: /* revfn3.c                                                           */
 3: /* Author: Bob Dondero                                                */
 4: /*--------------------------------------------------------------------*/
 5:
 6: #include <stdio.h>
 7: #include <assert.h>
 8:
 9: /*--------------------------------------------------------------------*/
10:
11: /* The number of elements in the array. */
12: enum {ARRAY_LENGTH = 5};
13:
14: /*--------------------------------------------------------------------*/
15:
16: /* piNums is a pointer to the 0th element of an array.  Read
17:    integers from stdin into that array.  */
18:
19: static void getNumbers(int piNums[])
20: {
21:    int i;
22:
23:    assert(piNums != NULL);
24:
25:    for (i = 0; i < ARRAY_LENGTH; i++)
26:       scanf("%d", &piNums[i]);
27: }
28:
29: /*--------------------------------------------------------------------*/
30:
31: /* piNums is a pointer to the 0th element of an array.  Print the
32:    elements of that array to stdout in reverse order.  */
33:
34: static void putNumbers(int piNums[])
35: {
36:    int i;
37:
38:    assert(piNums != NULL);
39:
40:    for (i = ARRAY_LENGTH-1; i >= 0; i--)
41:       printf("%d\n", piNums[i]);
42: }
43:
44: /*--------------------------------------------------------------------*/
45:
46: /* Read ARRAY_LENGTH integers from stdin, and write them in reverse
47:    order to stdout.  Return 0. */
48:
49: int main(void)
50: {
51:    int aiNums[ARRAY_LENGTH];
52:
53:    printf("Enter %d integers:\n", ARRAY_LENGTH);
54:    getNumbers(aiNums);
55:
56:    printf("\n");
57:    printf("The integers in reverse order are:\n");
58:    putNumbers(aiNums);
59:
60:    return 0;
61: }
```

Princeton University
COS 217: Introduction to Programming Systems
Manipulating C Strings

String Operation	String in Stack	String in Rodata Section

Allocating memory for a string	`{`
	`  char ac[5];`
	`  ...`
	`}`
		`{`
		`  ...`
		`  ..."hi"...`
		`  ...`
		`}`

Initializing a string	`{`
	`  char acA[3] = {'h', 'i', '\0'};`
	`  char acB[] = {'h', 'i', '\0'};`
	`  char acC[2] = {'h', 'i', '\0'}/*warning*/`
	`  char acD[10] = {'h', 'i', '\0'};`
	`  char acE[3] = "hi";`
	`  char acF[] = "hi";`
	`  char acG[2] = "hi"; /* no warning!!! */`
	`  char acH[10] = "hi";`
	`  ...`
	`}`
		`{`
		`  ...`
		`  ..."hi"...`
		`  ...`
		`}`

Computing the length of a string	`{`
	`  char ac[20] =`
	`  "hello, world";`
	
	`  /* Evaluates to 12 */`
	`  ... strlen(ac) ...`
	
	`  /* Evaluates to 20 */`
	`  ... sizeof(ac) ...`
	`}`
		`{`
		`  char *pc =`
		`  "hello, world";`
		
		`  /* Evaluates to 12 */`
		`  ... strlen(pc) ...`
		
		`  /* Evaluates to 8 */`
		`  ... sizeof(pc) ...`
		`}`

Changing the characters of a string	`{`
	`  char ac[10] = "hi";`
	
	`  /* Compile-time error. */`
	`  ac = "bye";`
	
	`  /* The long way. */`
	`  ac[0] = 'b';`
	`  ac[1] = 'y';`
	`  ac[2] = 'e';`
	`  ac[3] = '\0';`
	
	`  /* The shortcut. */`
	`  strcpy(ac, "bye");  /* Dangerous. */`
	`}`
		(Runtime error to attempt to change the characters of a string that resides in the rodata section)

Concatenating characters onto a string	`{`
	`  char ac[10] = "hi";`
	
	`  /* Compile-time error. */`
	`  ac += "bye";`
	
	`  /* The long way. */`
	`  ac[2] = 'b';`
	`  ac[3] = 'y';`
	`  ac[4] = 'e';`
	`  ac[5] = '\0';`
	
	`  /* The shortcut. */`
	`  strcat(ac, "bye");  /* Dangerous. */`
	`}`
		(Runtime error to attempt to change the characters of a string that resides in the rodata section)

Comparing one string with another	`{`
	`  char acA[] = "hi";`
	`  char acB[] = "bye";`
	
	`  /* Legal, but compares addresses!!! */`
	`  if (acA < acB) ...`
	
	`  /* Compares strings */`
	`  if (strcmp(acA, acB) < 0) ...`
	`}`
		(Same as string in stack)

Reading a string	`{`
	`  char ac[10];`
	
	`  /* Reads a word as a string. */`
	`  iConvCount = scanf("%s", ac);`
	`  **/* Dangerous. */**`
	
	`  /* Reads a line as a string,`
	`  removing the \n character. */`
	`  pcRet = gets(ac);`
	`  **/* Dangerous. */**`
	
	`  /* Reads a line as a string,`
	`  retaining the \n character. */`
	`  pcRet = fgets(ac, 10, stdin);`
	`}`
		(Runtime error to attempt to change the characters of a string that resides in the rodata section)

Writing a string	`{`
	`  char ac[] = "hi";`
	
	`  /* Writes a string. */`
	`  iCharCount = printf("%s", acStr);`
	
	`  /* Writes a string, appending a \n`
	`  character. */`
	`  iSuccessful = puts(ac);`
	
	`  /* Writes a string. */`
	`  iSuccessful = fputs(ac, stdout);`
	`}`
		(Same as string in stack)

Converting a string to another type	`{`
	`  char ac[] = "123";`
	`  int i;`
	`  long l;`
	`  double d;`
	`  iConvCount = sscanf(ac, "%d", &i);`
	`  i = atoi(ac);`
	`  l = atol(ac);`
	`  d = atof(ac);`
	`}`
		(Same as string in stack)

Converting another type to a string	`{`
	`  char ac[10];`
	`  int i = 123;`
	`  iCharCount = sprintf(ac, "%d", i);`
	`  **/* Dangerous. */**`
	`}`
		(Runtime error to attempt to change the characters of a string that resides in the rodata section)

Princeton University
COS 217: Introduction to Programming Systems
The `const` Keyword with Pointers

Pointer to Constant

```
1:  const int i1 = 100;
2:  const int i2 = 200;
3:  const int *pi = &i1;          /* pi is a "pointer to a constant." */
4:  i1 = 300;                     /* Error.  Cannot change i1. */
5:  i2 = 400;                     /* Error.  Cannot change i2. */
6:  pi = &i2;                     /* OK. */
7:  *pi = 500;                    /* Error.  Cannot change *pi. */
```

Constant Pointer

```
1:  int i1 = 100;
2:  int i2 = 200;
3:  int *const pi = &i1;          /* pi is a "constant pointer." */
4:  i1 = 300;                     /* OK. */
5:  i2 = 400;                     /* OK. */
6:  pi = &i2;                     /* Error.  Cannot change pi. */
7:  *pi = 500;                    /* OK. */
```

Constant Pointer to Constant

```
1:  const int i1 = 100;
2:  const int i2 = 200;
3:  const int *const pi = &i1;    /* pi is a "constant pointer to a constant." */
4:  i1 = 300;                     /* Error.  Cannot change i1. */
5:  i2 = 400;                     /* Error.  Cannot change i2. */
6:  pi = &i2;                     /* Error.  Cannot change pi. */
7:  *pi = 500;                    /* Error.  Cannot change *pi. */
```

Disallowed Mismatch

```
1:  const int i1 = 100;
2:  const int i2 = 200;
3:  int *pi = &i1;        /* Error.  Subversive.
                             Subsequently changing *pi would change i1. */
```

Disallowed Mismatch in Function Calls

```
1:  void f(int *pi)
2:  {
3:      ...
4:  }

    ...

5:  const int i1 = 5;
6:  const int *pi2 = &i1;
7:  f(pi2);               /* Error.  Subversive.
                             If f() changes *pi, then *pi2 also would change. */
```

Allowed Mismatch

```
1:  int i1 = 100;
2:  int i2 = 200;
3:  const int *pi = &i1;  /* OK, even though subsequently changing i1 would change *pi. */
4:  i1 = 300;             /* OK.  Also changes *pi. */
5:  i2 = 400;             /* OK. */
6:  pi = &i2;             /* OK, even though subsequently changing i2 would change *pi. */
7:  *pi = 500;            /* Error.  Cannot change *pi. */
```

Allowed Mismatch in Function Calls

```
1:  void f(const int *pi)
2:  {
3:      ...
4:  }

    ...

5:  int i1 = 5;
6:  int *pi2 = &i1;
7:  f(pi2);               /* OK.  *pi2 is protected against accidental change by f(). */
```