Special Data Types : Assigning Values to enum Names

Defining enum Data Types

```
1:  /* 18L01.c: Defining enum data types */
2:  #include <stdio.h>
3:  /* main() function */
4:  main()
5:  {
6:     enum language {human=100,
7:                    animal=50,
8:                    computer};
9:     enum days{SUN,
10:              MON,
11:              TUE,
12:              WED,
13:              THU,
14:              FRI,
15:              SAT};
16:
17:    printf("human: %d,  animal: %d,  computer: %d\n",
18:       human, animal, computer);
19:    printf("SUN: %d\n", SUN);
20:    printf("MON: %d\n", MON);
21:    printf("TUE: %d\n", TUE);
22:    printf("WED: %d\n", WED);
23:    printf("THU: %d\n", THU);
24:    printf("FRI: %d\n", FRI);
25:    printf("SAT: %d\n", SAT);
26:
27:    return 0;
28: }
```

Special Data Types : Assigning Values to enum Names

Computer Screen

```
human: 100,  animal: 50,  computer: 51
SUN: 0
MON: 1
TUE: 2
WED: 3
THU: 4
FRI: 5
SAT: 6
```

Special Data Types : Assigning Values to enum Names

Using the enum Data Type

```
1:  /* 18L02.c: Using the enum data type */
2:  #include <stdio.h>
3:  /* main() function */
4:  main()
5:  {
6:     enum units{penny = 1,
7:                nickel = 5,
8:                dime = 10,
9:                quarter = 25,
10:               dollar = 100};
11:    int money_units[5] = {
12:               dollar,
13:               quarter,
14:               dime,
15:               nickel,
16:               penny};
17:    char *unit_name[5] = {
18:              "dollar(s)",
19:              "quarter(s)",
20:              "dime(s)",
21:              "nickel(s)",
22:              "penny(s)"};
23:    int cent, tmp, i;
24:
```

```
25:    printf("Enter a monetary value in cents:\n");
26:    scanf("%d", &cent);  /* get input from the user */
27:    printf("Which is equivalent to:\n");
28:    tmp = 0;
29:    for (i=0; i<5; i++){
30:       tmp = cent / money_units[i];
31:       cent -= tmp * money_units[i];
32:       if (tmp)
33:         printf("%d %s ", tmp, unit_name[i]);
34:    }
35:    printf("\n");
36:    return 0;
37: }
```

Computer Screen

```
Enter a monetary value in cents:
141
Which is equivalent to:
1 dollar(s) 1 quarter(s) 1 dime(s) 1 nickel(s) 1 penny(s)
```

Special Data Types : Why Use typedef?

Using typedef Definitions

```
1:  /* 18L03.c: Using typedef definitions */
2:  #include <stdio.h>
3:  #include <stdlib.h>
4:  #include <string.h>
5:
6:  enum constants{ITEM_NUM = 3,
7:                  DELT='a'-'A'};
8:  typedef char *STRING[ITEM_NUM];
9:  typedef char *PTR_STR;
10: typedef char CHAR;
11: typedef int INTEGER;
12:
13: void Convert2Upper(PTR_STR str1, PTR_STR str2);
14:
15: main()
16: {
17:    STRING str;
18:    STRING moon = {"Whatever we wear",
19:                   "we become beautiful",
20:                   "moon viewing!"};
21:    INTEGER i;
22:    INTEGER term = 0;
23:
24:    for (i=0; i<ITEM_NUM; i++){
25:      str[i] = malloc((strlen(moon[i])+1) * sizeof(CHAR));
26:      if (str[i] == NULL){
27:        printf("malloc() failed.\n");
28:        term = 1;
29:        i = ITEM_NUM;  /* break the for loop */
30:      }
31:      Convert2Upper(moon[i], str[i]);
32:      printf("%s\n", moon[i]);
33:    }
34:    for (i=0; i<ITEM_NUM; i++){
35:      printf("\n%s", str[i]);
36:      free (str[i]);
37:    }
38:    printf("\n");
39:    return term;
40: }
41: /* function definition */
42: void Convert2Upper(PTR_STR str1, PTR_STR str2)
43: {
44:    INTEGER i;
45:
46:    for (i=0; str1[i]; i++){
47:      if ((str1[i] >= 'a') &&
48:          (str1[i] <= 'z'))
49:        str2[i] = str1[i] - DELT;
50:      else
51:        str2[i] = str1[i];
52:    }
53:    str2[i] = '\0';  /* add null character */
54: }
```

Special Data Types : Why Use typedef?

Computer Screen

Recursive Functions

Calling a Recursive Function

```
 1:  /* 18L04.c: Calling a recursive function */
 2:  #include <stdio.h>
 3:
 4:  enum con{MIN_NUM = 0,
 5:           MAX_NUM = 100};
 6:
 7:  int fRecur(int n);
 8:
 9:  main()
10: {
11:    int i, sum1, sum2;
12:
13:    sum1 = sum2 = 0;
14:    for (i=1; i<=MAX_NUM; i++)
15:      sum1 += i;
16:    printf("The value of sum1 is %d.\n", sum1);
17:    sum2 = fRecur(MAX_NUM);
18:    printf("The value returned by fRecur() is %d.\n", sum2);
19:
20:    return 0;
21: }
22: /* function definition */
23: int fRecur(int n)
24: {
25:    if (n == MIN_NUM)
26:      return 0;
27:    return  fRecur(n - 1) + n;
28: }
```

Computer Screen

```
The value of sum1 is 5050.
The value returned by fRecur() is 5050.
```

Command-Line Arguments

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

   printf("The value received by argc is %d.\n", argc);
   printf("There are %d command-line arguments passed to main().\n", argc);
   if(argc) {
         printf("The first command-line argument is: %s\n", argv[0]);
         printf("The rest of the command-line arguments are:\n");
         for (i=1; i<argc; i++)
            printf("%s\n", argv[i]);
   }
   return 0;
}
```

Command-Line Arguments

Computer Screen

```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\Murat ŞİMŞEK>cd C:\Dev-Cpp\Examples\lecture_code

C:\Dev-Cpp\Examples\lecture_code> main_argument.exe hello world
The value received by argc is 3.
There are 3 command-line arguments passed to main().
The first command-line argument is: main_argument.exe
The rest of the command-line arguments are:
hello
world
```