02_10_ C enum

The **`enum`** Data Type

- The C language provides you with an additional data type—the **`enum`** data type. **`enum`** is short for enumerated.

- The enumerated data type can be used to declare named integer constants.

- The **`enum`** data type makes the C program more readable and easier to maintain.

Declaring the **enum** Data Type

- The general form of the **`enum`** data type declaration is

```
enum tag_name {enumeration_list} variable_list;
```

- Here **`tag_name`** is the name of the enumeration.
- **`variable_list`** gives a list of variable names that are of the **`enum`** data type.
- **`enumeration_list`** contains defined enumerated names that are used to represent integer constants.
- (Both **`tag_name`** and **`variable_list`** are optional.)

Declaring the **enum** Data Type

- For instance, the following declares an enum data type with the tag name of automobile:

```
enum automobile {sedan, pick_up, sport_utility};
```

- Given this, you can define enum variables like this:

```
enum automobile  domestic, foreign;
```

- Here the two enum variables, **`domestic`** and **`foreign`**, are defined.

Declaring the **enum** Data Type

- Of course, you can always declare and define a list of enum variables in a single statement, as shown in the general form of the enum declaration.

- Therefore, you can rewrite the enum declaration of domestic and foreign like this:

```
enum automobile {sedan, pick_up, sport_utility} domestic, foreign;
```

Declaring the enum Data Type

```
enum {
     DOG,
     CAT,
     FISH,
};
```

```
enum animals {
     DOG,
     CAT,
     FISH,
};
```

```
enum animals {
     DOG,
     CAT,
     FISH,
} pet;
```

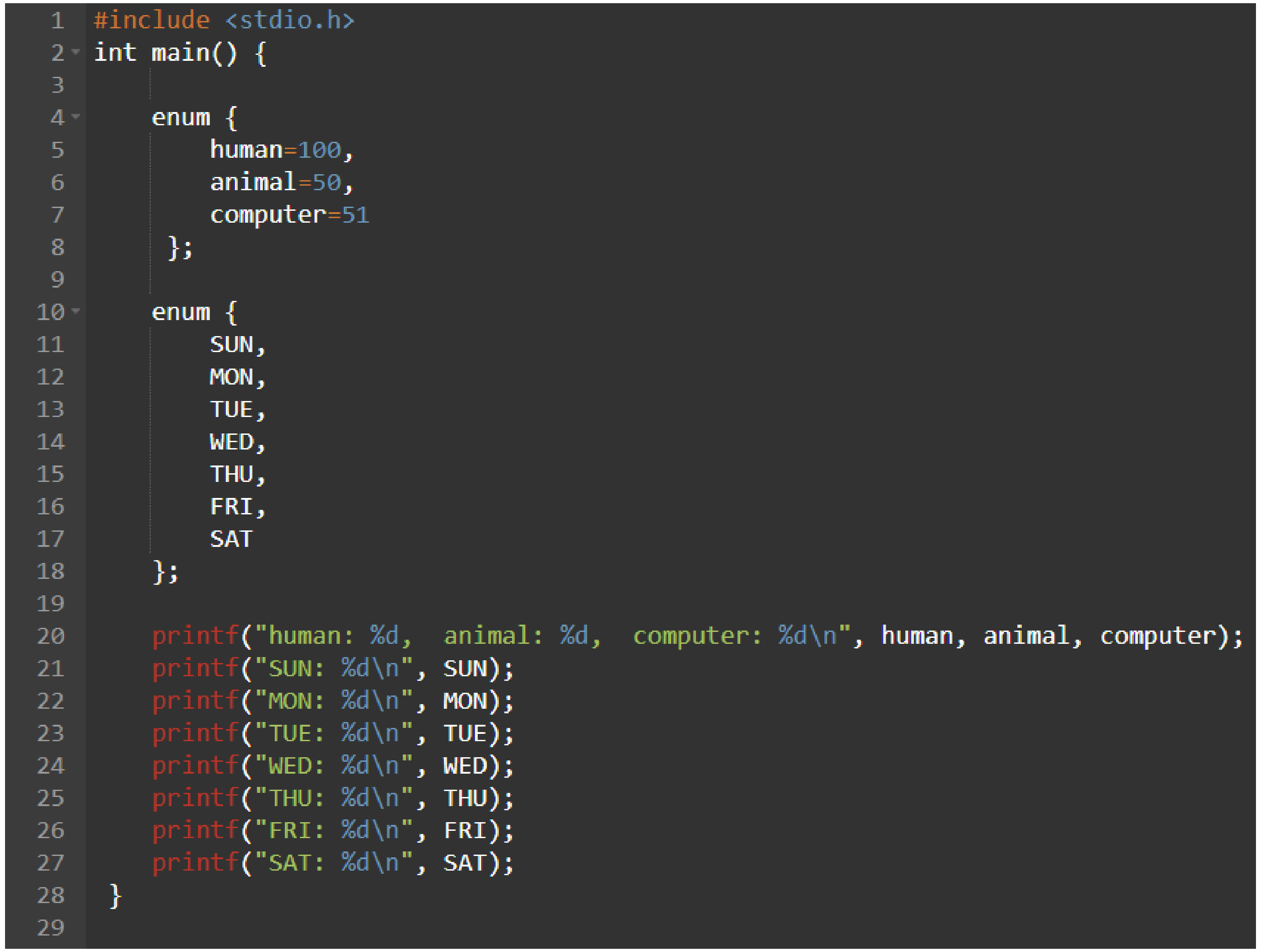
```
enum {
     CAR,
     BUS,
     TRAIN,
};
```

```
enum transport {
     CAR,
     BUS,
     TRAIN,
};
```

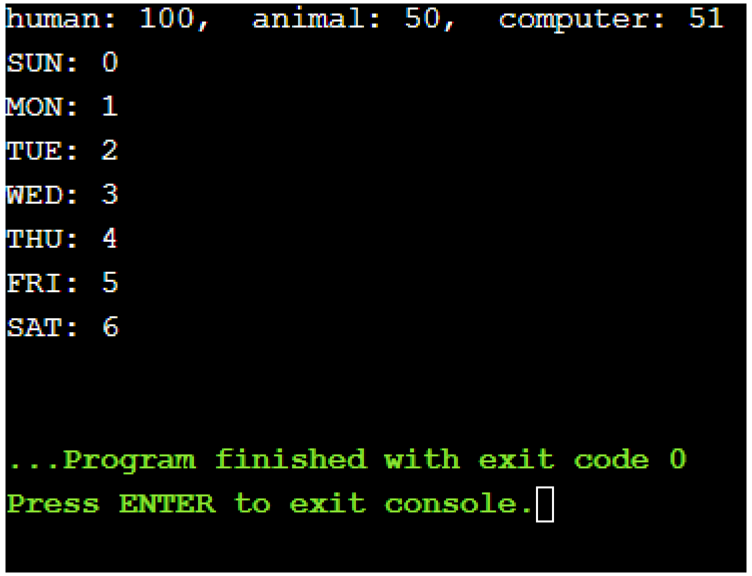
```
enum transport {
     CAR,
     BUS,
     TRAIN,
} vehicle;
```

```
void foo(enum animals jerry);
```

Declaring the **enum** Data Type No name

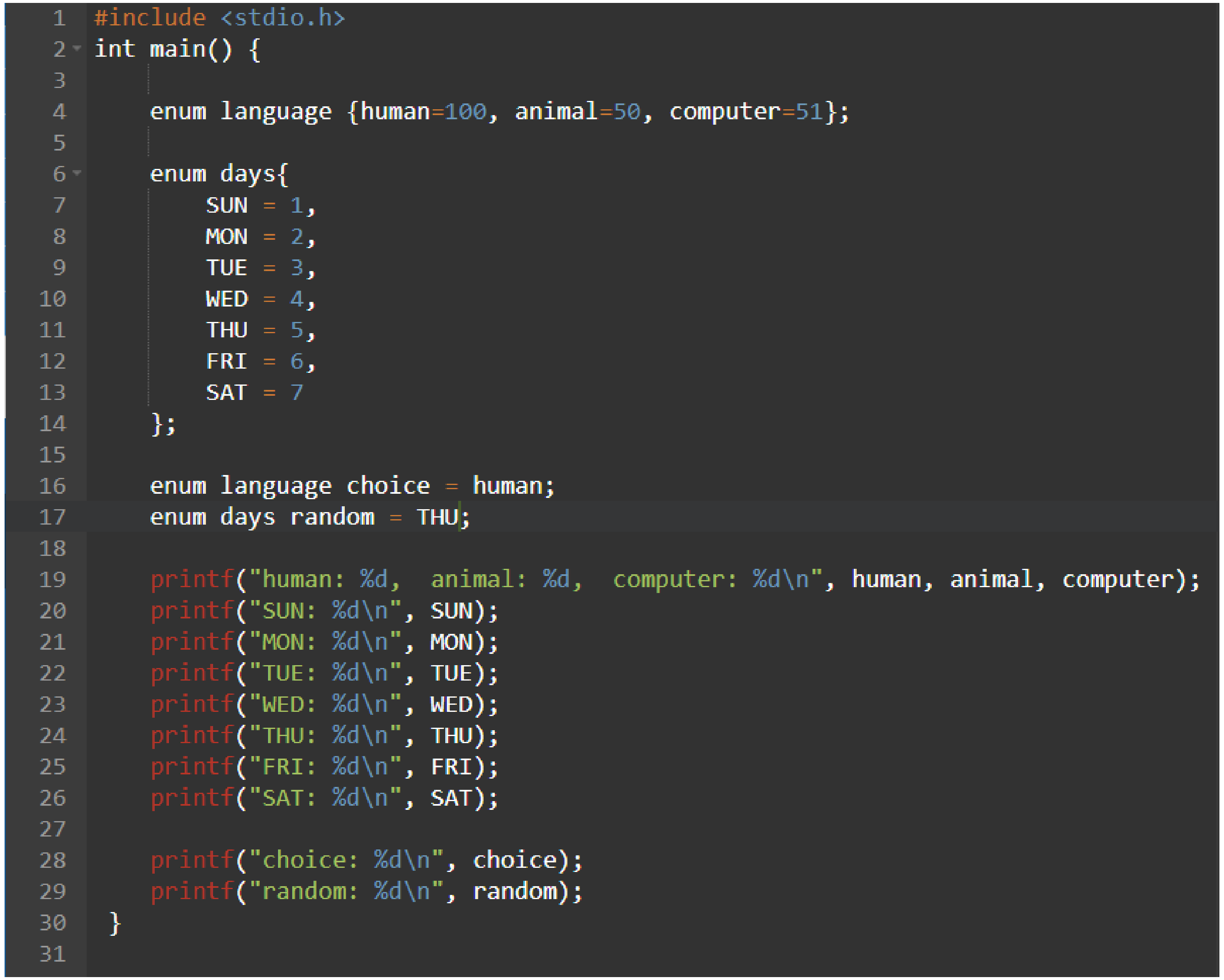


```c
#include <stdio.h>
int main() {
    
    enum {
        human=100,
        animal=50,
        computer=51
     };
    
    enum {
        SUN,
        MON,
        TUE,
        WED,
        THU,
        FRI,
        SAT
    };

    printf("human: %d,  animal: %d,  computer: %d\n", human, animal, computer);
    printf("SUN: %d\n", SUN);
    printf("MON: %d\n", MON);
    printf("TUE: %d\n", TUE);
    printf("WED: %d\n", WED);
    printf("THU: %d\n", THU);
    printf("FRI: %d\n", FRI);
    printf("SAT: %d\n", SAT);
 }
 
```

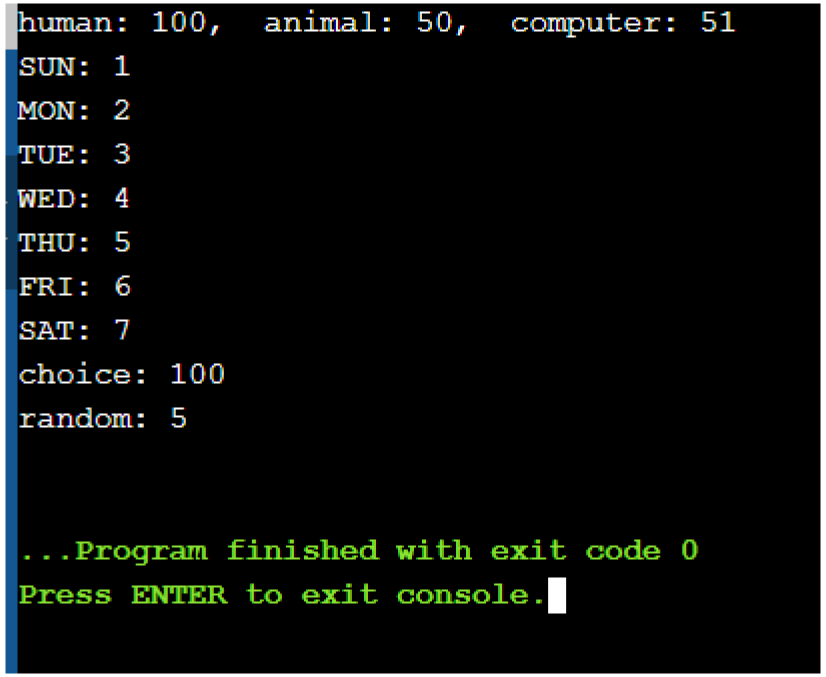


```
human: 100,  animal: 50,  computer: 51
SUN: 0
MON: 1
TUE: 2
WED: 3
THU: 4
FRI: 5
SAT: 6


...Program finished with exit code 0
Press ENTER to exit console.
```

Declaring the **enum** Data Type with Name

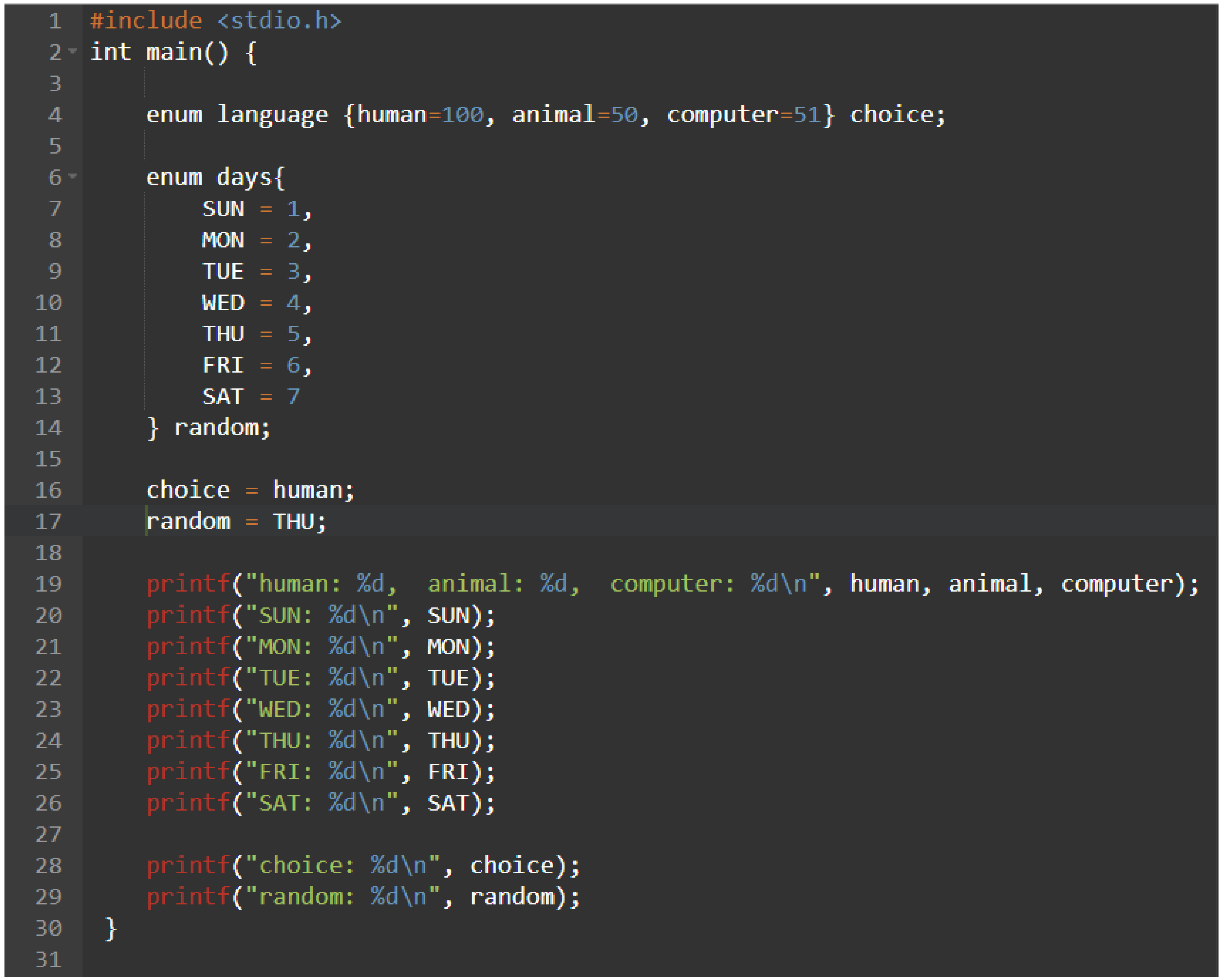


```c
#include <stdio.h>
int main() {

    enum language {human=100, animal=50, computer=51};

    enum days{
        SUN = 1,
        MON = 2,
        TUE = 3,
        WED = 4,
        THU = 5,
        FRI = 6,
        SAT = 7
    };

    enum language choice = human;
    enum days random = THU;

    printf("human: %d,  animal: %d,  computer: %d\n", human, animal, computer);
    printf("SUN: %d\n", SUN);
    printf("MON: %d\n", MON);
    printf("TUE: %d\n", TUE);
    printf("WED: %d\n", WED);
    printf("THU: %d\n", THU);
    printf("FRI: %d\n", FRI);
    printf("SAT: %d\n", SAT);

    printf("choice: %d\n", choice);
    printf("random: %d\n", random);
 }

```

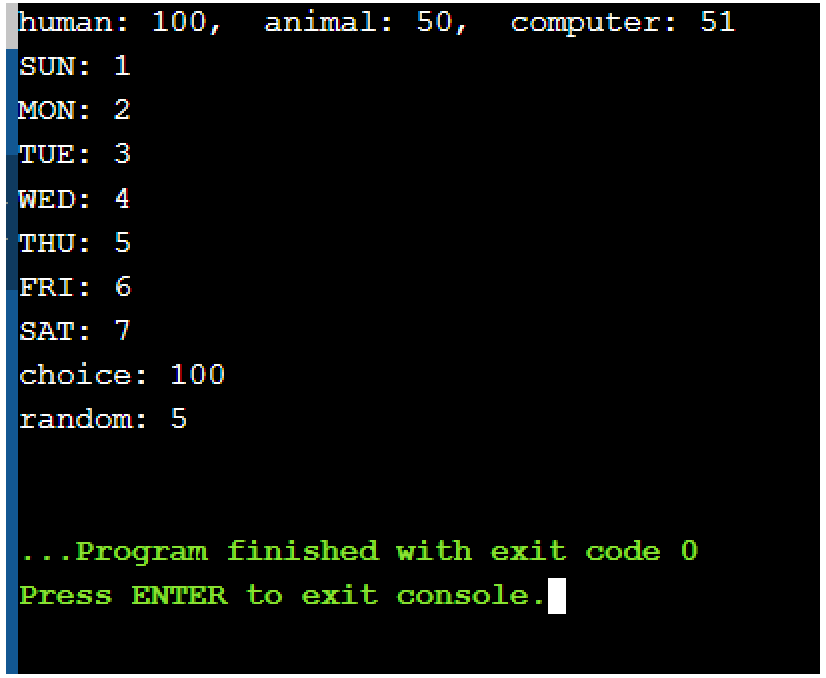


```
human: 100,  animal: 50,  computer: 51
SUN: 1
MON: 2
TUE: 3
WED: 4
THU: 5
FRI: 6
SAT: 7
choice: 100
random: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Declaring the **enum** Data Type with Name



```c
#include <stdio.h>
int main() {

    enum language {human=100, animal=50, computer=51} choice;

    enum days{
        SUN = 1,
        MON = 2,
        TUE = 3,
        WED = 4,
        THU = 5,
        FRI = 6,
        SAT = 7
    } random;

    choice = human;
    random = THU;

    printf("human: %d,  animal: %d,  computer: %d\n", human, animal, computer);
    printf("SUN: %d\n", SUN);
    printf("MON: %d\n", MON);
    printf("TUE: %d\n", TUE);
    printf("WED: %d\n", WED);
    printf("THU: %d\n", THU);
    printf("FRI: %d\n", FRI);
    printf("SAT: %d\n", SAT);

    printf("choice: %d\n", choice);
    printf("random: %d\n", random);
 }

```



```
human: 100,  animal: 50,  computer: 51
SUN: 1
MON: 2
TUE: 3
WED: 4
THU: 5
FRI: 6
SAT: 7
choice: 100
random: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

Assigning Values to enum Names

- By default, the integer value associated with the leftmost name in the enumeration list field, surrounded by the braces (**`{`** and **`}`**), starts with **`0`**, and the value of each name in the rest of the list increases by one from left to right. Therefore, in the previous example, **`sedan`**, **`pick_up`**, and **`sport_utility`** have the values of **`0`**, **`1`**, and **`2`**, respectively.

- In fact, you can assign integer values to **`enum`** names.
- Considering the previous example, you can initialize the enumerated names like :

```
enum automobile {sedan = 60, pick_up = 30, sport_utility = 10};
```

- Now, **`sedan`** represents the value of **`60`**, **`pick_up`** has the value of **`30`**, and **`sport_utility`** assumes the value of **`10`**.

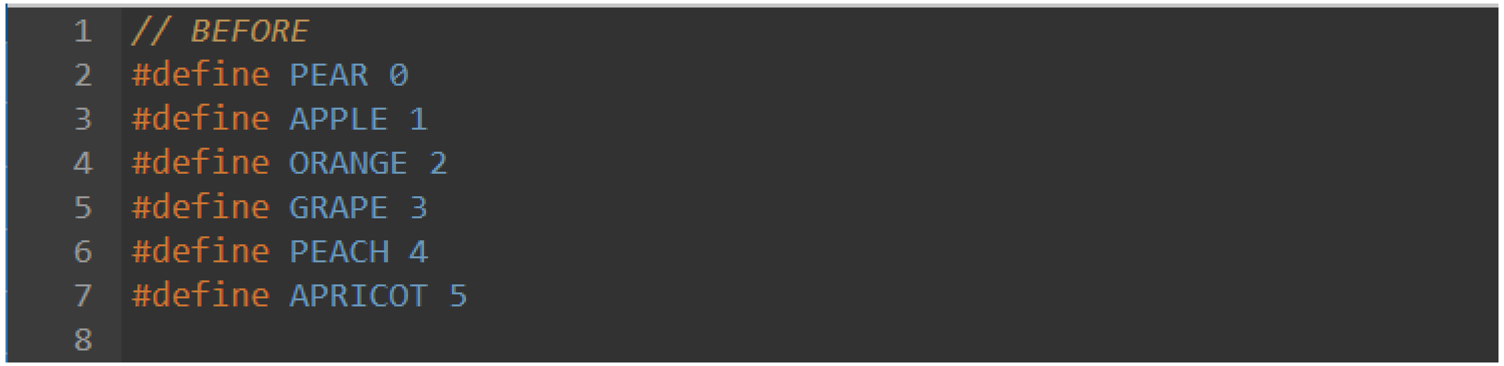
```c
/* 20L02.c: Using the enum data type */
#include <stdio.h>
/* main() function */
main()
{
   enum units{penny = 1,
              nickel = 5,
              dime = 10,
              quarter = 25,
              dollar = 100};
   int money_units[5] = {
              dollar,
              quarter,
              dime,
              nickel,
              penny};
   char *unit_name[5] = {
             "dollar(s)",
             "quarter(s)",
             "dime(s)",
             "nickel(s)",
             "penny(s)"};
   int cent, tmp, i;

   printf("Enter a monetary value in cents:\n");
   scanf("%d", &cent);  /* get input from the user */
   printf("Which is equivalent to:\n");
   tmp = 0;
   for (i=0; i<5; i++){
      tmp = cent / money_units[i];
      cent -= tmp * money_units[i];
      if (tmp)
        printf("%d %s ", tmp, unit_name[i]);
   }
   printf("\n");
   return 0;
}
```

```
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)
```

Why use **enum** when **#define** is just as efficient?

- The advantages of **enum** show up when you have a long list of things you want to map into numbers, and you want to be able to insert something in the middle of that list.

- for example, you have:

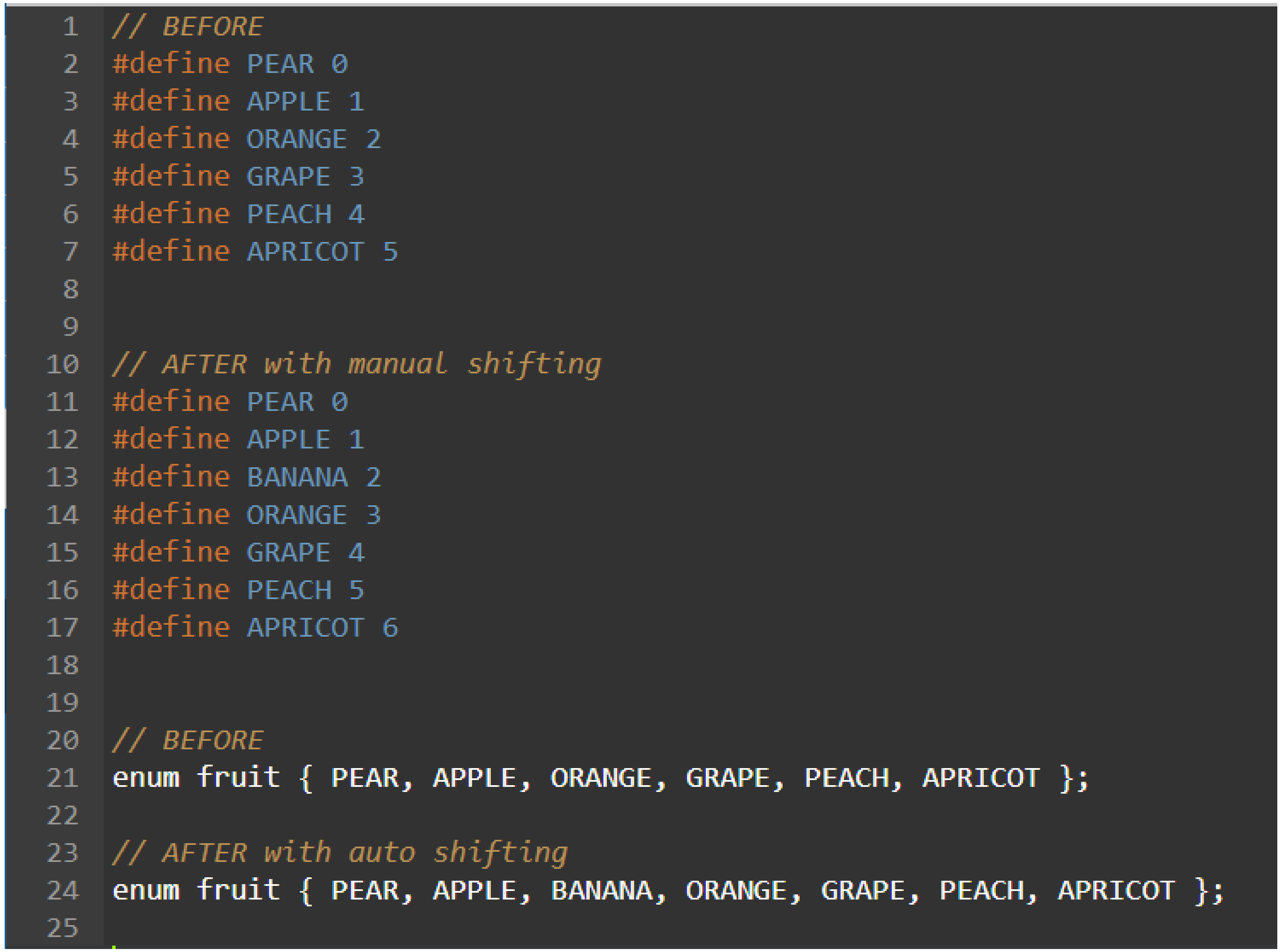


```
// BEFORE
#define PEAR 0
#define APPLE 1
#define ORANGE 2
#define GRAPE 3
#define PEACH 4
#define APRICOT 5

```

- now you want to put banana after oranges. with `#define`, you'd have to redefine the numbers of grapes, peaches, and apricots.

- using `enum`, it would happen automatically.

Why use **enum** when **#define** is just as efficient?



```
// BEFORE
#define PEAR 0
#define APPLE 1
#define ORANGE 2
#define GRAPE 3
#define PEACH 4
#define APRICOT 5


// AFTER with manual shifting
#define PEAR 0
#define APPLE 1
#define BANANA 2
#define ORANGE 3
#define GRAPE 4
#define PEACH 5
#define APRICOT 6


// BEFORE
enum fruit { PEAR, APPLE, ORANGE, GRAPE, PEACH, APRICOT };

// AFTER with auto shifting
enum fruit { PEAR, APPLE, BANANA, ORANGE, GRAPE, PEACH, APRICOT };

```

End of 02_10