Arrays, Formatted I/O, and Flow Control

Unit 2 Objectives

- Declare and use arrays
- Use printf() and scanf() functions for writing to *standard output*, reading from *standard input*
- Implement C flow control statements

  if, while (Unit 1)
  do-while
  for
  switch
  continue
  break
  goto

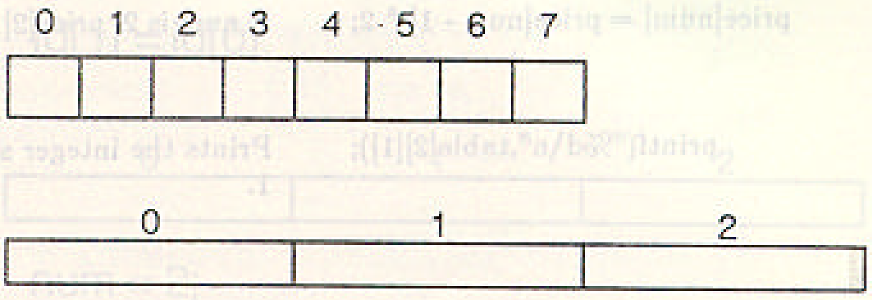
Arrays

Arrays

- Collection of Identically Typed Objects
- Single or Multi-dimensional
- Sequential Storage
- Index: 0 to (size - 1)

char id[8];

float price[3];



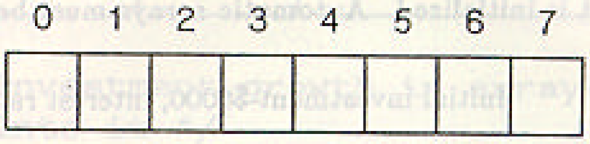
int table[4][3];

	col0	*col1*	*col2*
row0	0 0	0 1	0 2
row1	1 0	1 1	1 2
row2	2 0	2 1	2 2
row3	3 0	3 1	3 2

Array Element Reference

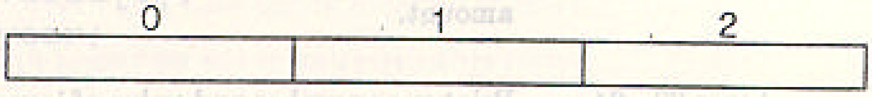
array_name[*integer expression*]

char id[8];



```
id[0] = 'j';
id[1] = id[0];
```

float price[3];



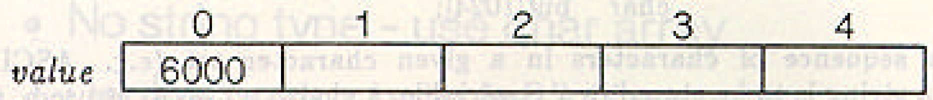
```
num = 2;
price[num] = price[num - 1] * 2;
```

int table[4][3];

	col0	*col1*	*col2*
row0	0 0	0 1	0 2
row1	1 0	1 1	1 2
row2	2 0	2 1	2 2
row3	3 0	3 1	3 2

```
printf("%d\n",table[2][1]);
```

Arrays - Sample Program

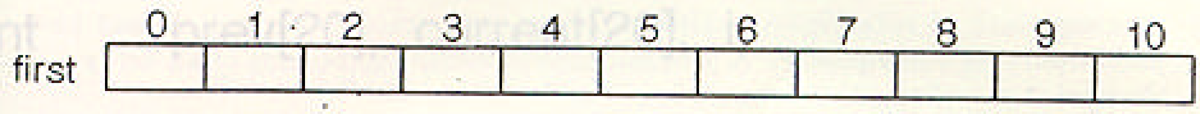


```c
/* Stores investment growth in array */
/* and prints it */
#include    <stdio.h>
main()
{
    float  investment = 6000;
    float  interest = .085, rate;
    double value[5];
    int    year;

    rate = 1 + interest;  /*108.5% of prev year*/
    value[0] = investment;
    year = 1;
    while (year < 5) {
           value[year] = value[year - 1] * rate;
           year += 1;   /* year = year + 1 */
           }
    printf("Initial investment: $%.2f\n",value[0])
    year = 1;
    while (year < 5) {  /*print 2 decimal places*/
           printf("Year %d: ",year);
           printf("$%.2f\n",value[year]);
           year += 1;   /* year = year + 1 */
           }
}
```

Character Arrays

- No string type - use char array
- Declare extra byte for '\0' at end



```
/* Prompts user for name and echoes it */
#include <stdio.h>
main()
{
  int     i;
  char    first[11];   /*allow 10 char name*/

  printf("Please enter first name: ");
  i = 0;
  while (i < 10 && (first[i] = getchar())!= '\n')
         i += 1;
  first[i] = '\0';
  /*%s needs addr of null-terminated string*/
  printf("Name entered: %s\n",&first[0]);
  /* printf("Name entered: %s\n",first);*/
}
```

```
NOTE: Easier ways of reading char arrays
      are discussed later.
```

Copying Arrays

☞ **Arrays must be copied element by element** ☜

```
int     prev[20],  current[20],  i;
```

Incorrect:

```
        prev = current;   /* Compile error */
```

Correct:

```
        i = 0;
        while (i < 20) {
                prev[i] = current[i];
                i += 1;
                }
```

Overrunning Arrays

For execution speed, C does not check array subscripts

+ Faster execution

– May read/write beyond array

– Program may abort

What are the valid subscripts for this array?

```
char    line[256];
```

What's Wrong Here?

```
int    ray[20], i;

i = 0;
while ( i <= 20 ) {
       ray[i] = i * 10;
       i += 1;
       }
```

Increment and Decrement Operators

- Concise and efficient; often used with arrays

x = x + 1;	*prefix:*	++x;
	postfix:	x++;
x = x − 1;	*prefix:*	−−x;
	postfix:	x−−;

prefix:	Changes lvalue immediately.
postfix:	Changes lvalue after it is used.

x = 3;
y = ++x;
	x = 3;
	y = x++;
		x = 3;
		y = −−x;
			x = 3;
			y = x−−;

y __________	y __________	y __________	y __________
x __________	x __________	x __________	x __________

Example:
```
i = 0;
while (i < size) {
      table[i] = 0;
      i = i + 1;
      }
```

Equivalent:
```
i = 0;
while (i < size)
      table[i++] = 0;
```

Formatted I/O

printf() and scanf()

Formatted I/O Overview
printf() and scanf()

- Standard I/O Library functions
- Workhorses for formatted I/O

	writing	reading
terminal*	printf()	scanf()
file	fprintf()	fscanf()
string	sprintf()	sscanf()

* standard input/output

printf() conversion characters

%conversion-char	
c	character
d,o,u,x,X	integer: decimal,octal,unsigned,hex (a-f or A-F) ld, lo, lu, lx, lX if type long
e,E	floating point - scientific notation
f	floating point - decimal notation
s	string ('\0' terminated)
%	literal %

```
#include <stdio.h>
main()
{
     char c = 'j';
     int val = 59;
     float total = 7500.5;
                                        Output:
     printf("%c\n",c);                  j
     printf("%d\n",val);                59
     printf("%o\n",val);                73
     printf("%x\n",val);                3b
     printf("%e\n",total);              7.500500e+03
     printf("%f\n",total);              7500.500000
}
```

printf() - additional formatting

	Meaning	Examples, Comments
num	minimum field width	right justified, leading blanks. %15d,%15s field at least 15 chars
.num	precision	%2f two places after decimal %15s at most 15 chars
-	left justify	%-6d left justify, min. field 6

Example 1: No field widths used:

```
printf("%d %d %s %f\n",mod,qt,&it[0],cst);

2901 6 Cerebral Calculator 75.489998
30 7229 Blue Ribbon Cable 26.000000
31650 100 Glow Worm Glare Screen 89.989998
2 677 Personal Mainframe 9000.000000
```

Example 2: Field widths used:

```
printf("%-6d %4d %-24s %7.2f\n",mod,qt,&it[0],cst);

2901      6 Cerebral Calculator        75.49
30     7229 Blue Ribbon Cable          26.00
31650   100 Glow Worm Glare Screen     89.99
2       677 Personal Mainframe       9000.00
```

Exercise – printf()

Do these exercises on paper (not on terminal). Given:

```
#include  <stdio.h>
main()
{
        int     val;
        float  sum;
        char  name[36];
        .
        .
```

1. Write printf() statements that print
   a. the value in val : ______________________________
   b. the string in name : ______________________________
   c. the value of sum / val : ______________________________
2. Write a printf() statement that prints one line with
   • val in a left-justified column at least 8 characters wide,
   • sum in a right-justified column at least 10 characters wide with 3 decimal places:

   ____

Introduction to the Address Operator

- & is the address operator
- It provides the address of an object

```
/* Declare x and its value */
         int x = 3;
```

- Now the expression

  x

  refers to the *value* of x

- And the expression

  &x

  refers to the *address* of x

scanf()

NAME scanf

SYNOPSIS
```
#include    <stdio.h>
int scanf(format[,pointer list])
```

DESCRIPTION Reads characters from *stdin* according to *format*. Stops on first conflict, offending character left unread. Stores results at addresses in pointer list. Returns number of %'s matched, EOF on end-of-file.

%conversion-char

c	any 1 character
d,u,o,x,X	integer: decimal,unsigned,octal,hex precede with l long, h if short
e,f	float, precede with l if double
s	string of non-whites, '\0' added to destination string or array

EXAMPLE
```
#include      <stdio.h>
main()
{
   int    ret,num;

   printf("Please enter an integer: ");
   ret = scanf("%d",&num);
   ...
```

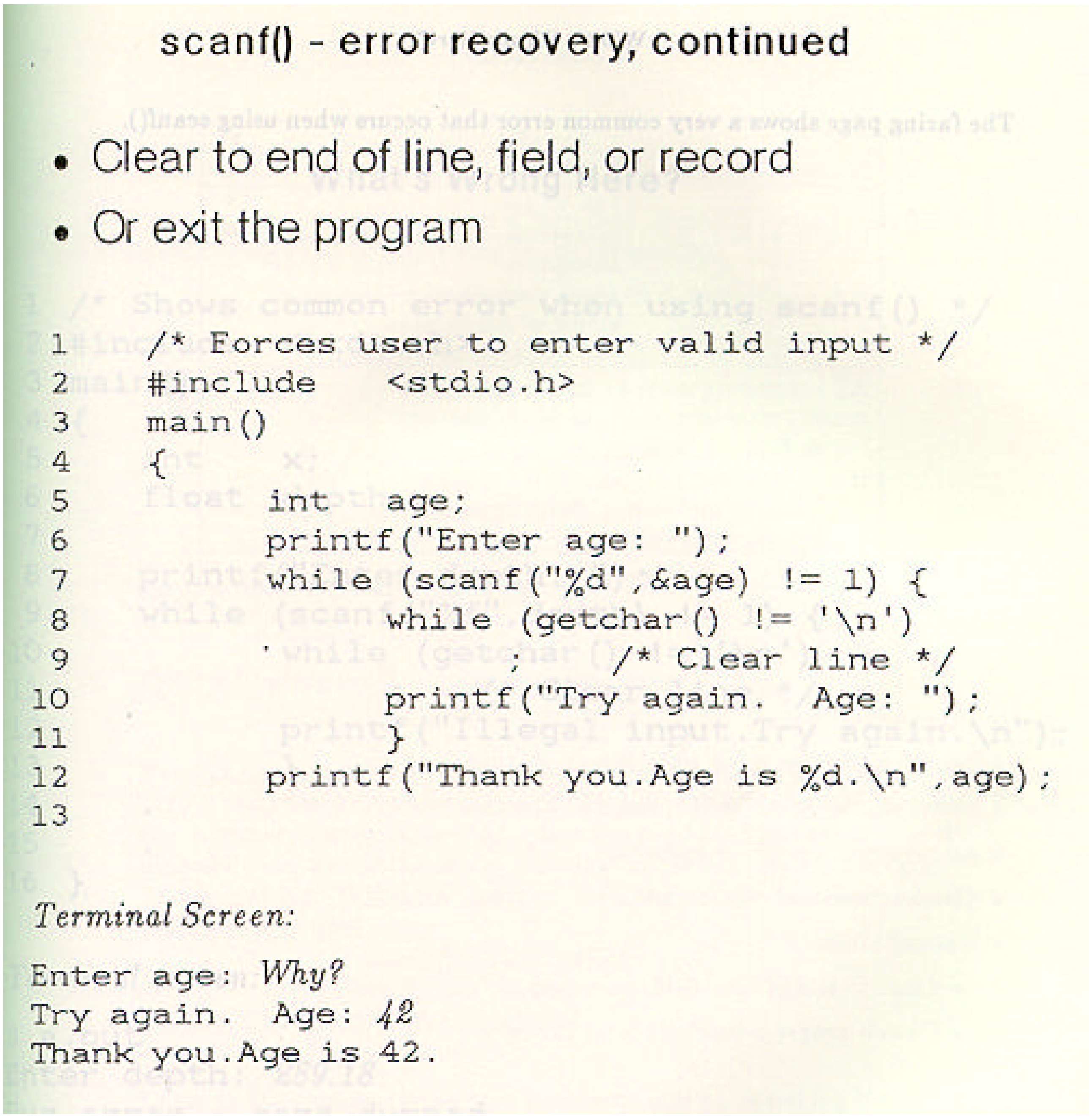
scanf() - error recovery

```c
/* Unsuccessful attempt to force*/
/* user to enter valid input    */
#include <stdio.h>
main()
{
   int   age;

   printf("Enter age: ");
   while  (scanf("%d",&age)  != 1)
         printf("Try again.  Age: ");
   printf("Thank you.Age is %d.\n",age);
   ...
```

Terminal Screen:

```
Enter age:  Why?
Try again.  Age: Try again.  Age:  ...
```



scanf() - error recovery, continued

- Clear to end of line, field, or record
- Or exit the program

```
/* Forces user to enter valid input */
#include   <stdio.h>
main()
{
          int    age;
          printf("Enter age: ");
          while (scanf("%d",&age) != 1) {
               while (getchar() != '\n')
                    ;    /* Clear line */
               printf("Try again.  Age: ");
               }
          printf("Thank you.Age is %d.\n",age);

```

Terminal Screen:

```
Enter age: Why?
Try again.  Age: 42
Thank you.Age is 42.
```

What's Wrong Here?

```
/* Shows common error when using scanf() */
#include  <stdio.h>
main()
{
    int    x;
    float  depth;

    printf("Enter depth: ");
    while (scanf("%f",depth) != 1) {
           while (getchar() != '\n')
                 ;   /* Clear line */
           printf("Illegal input.Try again.\n");
           }
    .
    .
}
```

Terminal screen:

```
$ a.out
Enter depth: 289.18
Bus error - core dumped
$
```

Exercise – scanf()

Do these exercises on paper, not on a terminal. For problems 1 and 2, use scanf() statements. Store the return value in the integer ‘x’.

```
#include  <stdio.h>

main()
{
        int         x,  sum ;
        double      diameter ;
        char        name[81] ;
        .
        .
```

1. Read a decimal integer into sum.

2. Read a floating point number into diameter.

3. The %s conversion character is used to read a string of characters into an array. Any leading white space on the input stream is skipped over. Then any number of non-white characters are read into the array. An ‘\0’ is placed at the end of the string. Assume the input stream contains “Teletype5620 WE32100 Layers”. If either of the following scanf() statements is used,

   ```
   scanf(“%s”, &name[0]);

   scanf(“%s”, name);
   ```

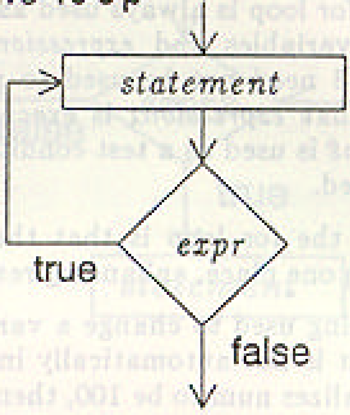
   what will the ‘name’ array contain?

   Later in the course, the library functions (gets(), fgets()) which read an entire line into an array are discussed.

Flow Control Statements

do-while loop

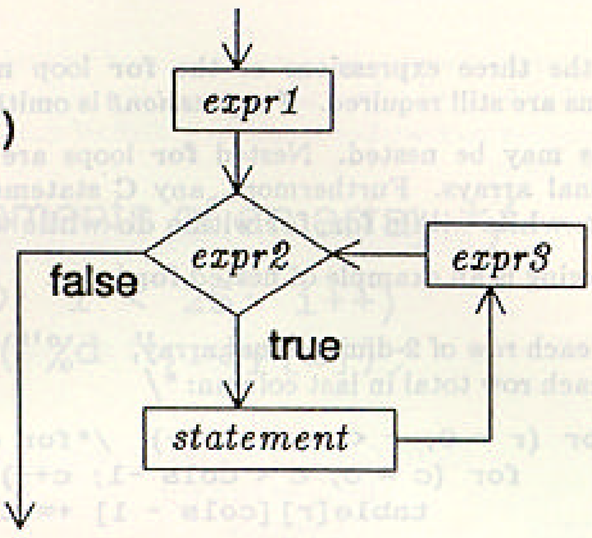
do
    statement
while (*expression*);



```
#include    <stdio.h>
main()     /* Do print menu while choice is not 'q' */
{
      char   choice;
      do {
             printf("\tMAIN MENU\n");
             printf("d - data entry\n");
             printf("r - report\n");
             printf("q - quit\n");
             scanf("%c",&choice);
             if (choice == 'd')
                   code for getting data entry here
             else if (choice == 'r')
                   code for printing report here
             else if (choice != 'q')
                         printf("Illegal choice.Try again.\n");
             while (getchar() != '\n')
                         ;   /* Clear line */
             }
      while (choice != 'q');
}
```

for loop

for (*expr1; expr2; expr3*)
    statement



Sample for loop:

```
for (i = 0; i < max; i++) {
    statement
    statement
    statement
}
```

Alternative using while:

```
i = 0;
while ( i < max ){
    statement
    statement
    statement
    i++;
}
```

for loop, continued

```
/* prints 25 elements of an array */
    for (i = 0; i < 25; i++)
        printf("%d ",ray[i]);


/* Sums elements in an array: */

    sum = 0;
    for (i = 0; i < max; )
        sum += quantity[i++];


/* Infinite loop: */

    for ( ; ; )
        doinit();
```

comma operator ,

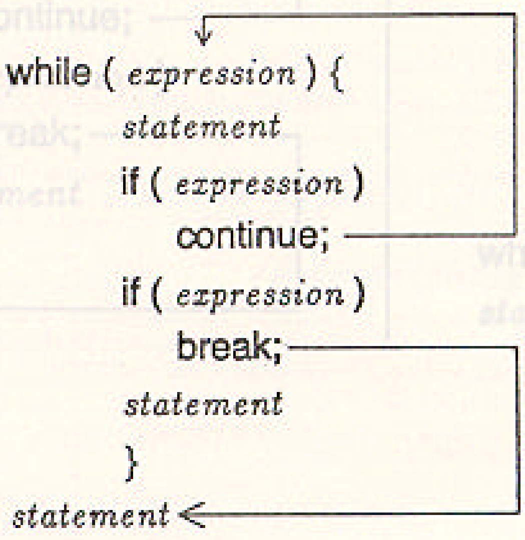
$$exprA,\ exprB$$

- First evaluate $exprA$, then $exprB$

```
                expr1              expr2        expr3
for ( i = 0, j = 10 ; i < max; i++, j++ )
              list[i] = name[j];
```

break and continue statements

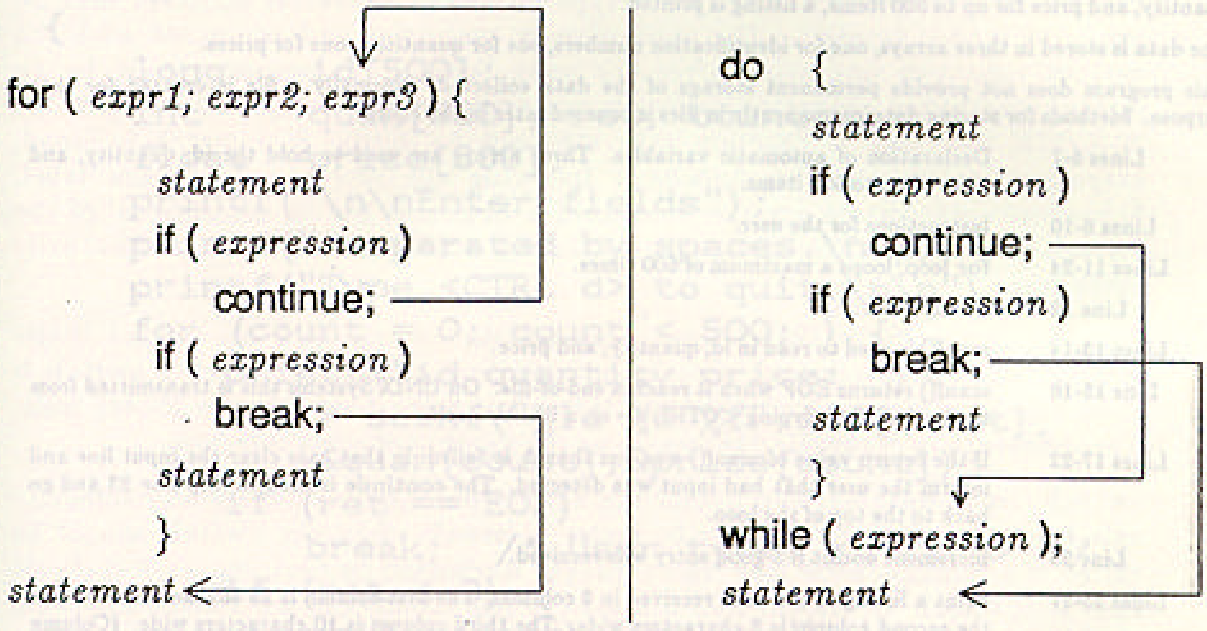
break	Causes immediate exit from innermost loop (while, do-while, for) or switch*
continue	Causes next iteration of innermost loop (while, do-while, for) to begin.



```
while ( expression ) {
    statement
    if ( expression )
        continue;
    if ( expression )
        break;
    statement
    }
statement
```

* Covered later in this unit

break and continue, continued

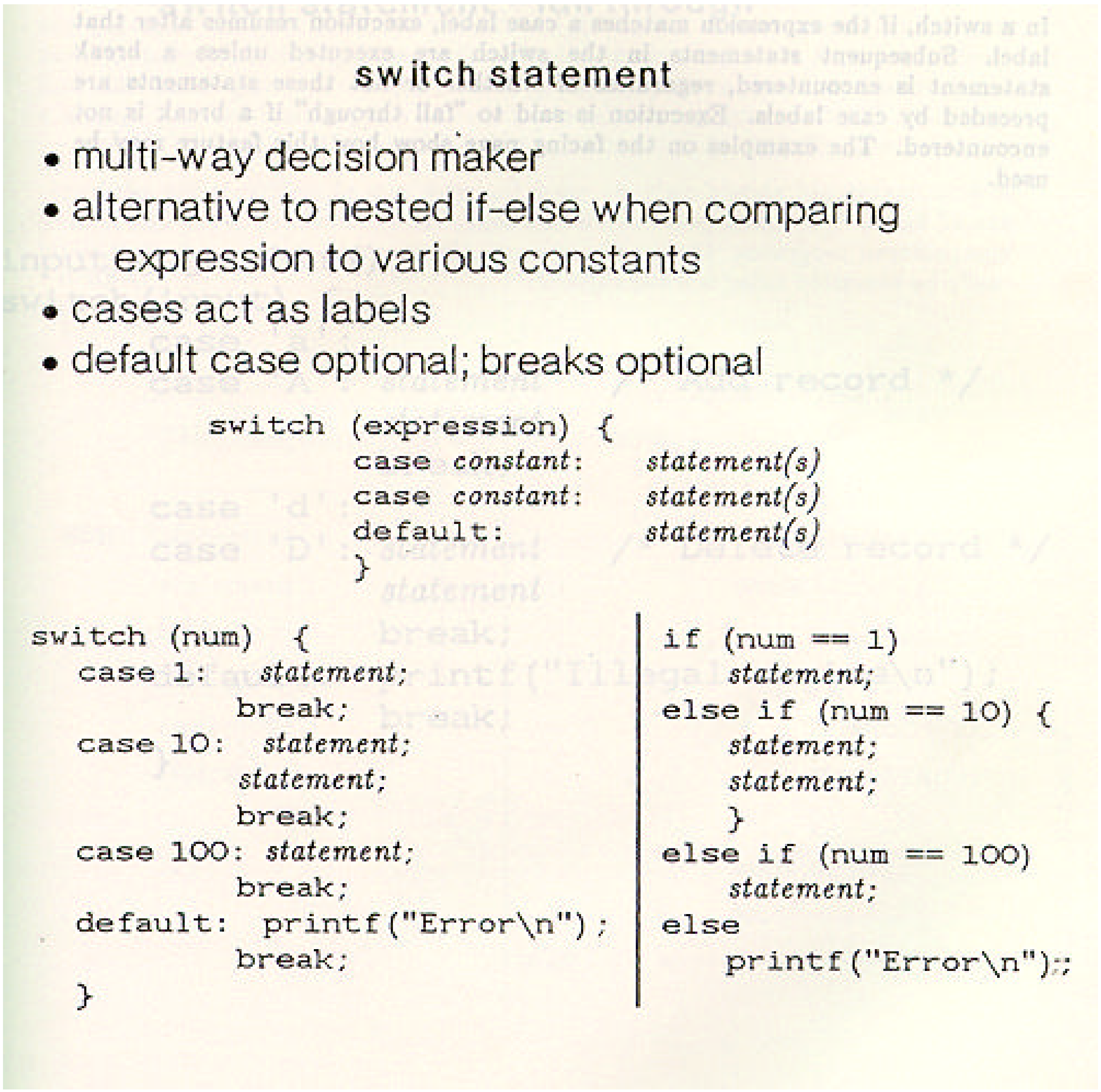


```
for ( expr1; expr2; expr3 ) {
        statement
        if ( expression )
            continue;
        if ( expression )
            break;
        statement
        }
statement
```

```
do   {
     statement
     if ( expression )
         continue;
     if ( expression )
         break;
     statement
     }
while ( expression );
statement
```

Sample Program

```
/*Data entry for inventory prog*/
#include    <stdio.h>
main()
{
    long    id[500];
    int     quan[500], ret, count, i;
    float   price[500];
    printf("\n\nEnter fields");
    printf(" separated by spaces.\n");
    printf("Type <CTRL d> to quit.\n\n");
    for (count = 0; count < 500; ) {
        printf("id quantity price: ");
        ret = scanf("%ld %d %f",&id[count],
            &quan[count],&price[count]);
        if (ret == EOF)
            break;  /* User typed <ctrl-d> */
        if (ret < 3) {
            while (getchar() != '\n')
                  ;     /* Clear line */
            printf("\tBad input, try again.\n");
            continue;
            }
        count++;
        }
    for (i = 0; i < count; i++)
            printf("\n%25ld %8d %10.2f\n",
                  id[i],quan[i],price[i]);
}
```



switch statement

- multi-way decision maker
- alternative to nested if-else when comparing expression to various constants
- cases act as labels
- default case optional; breaks optional

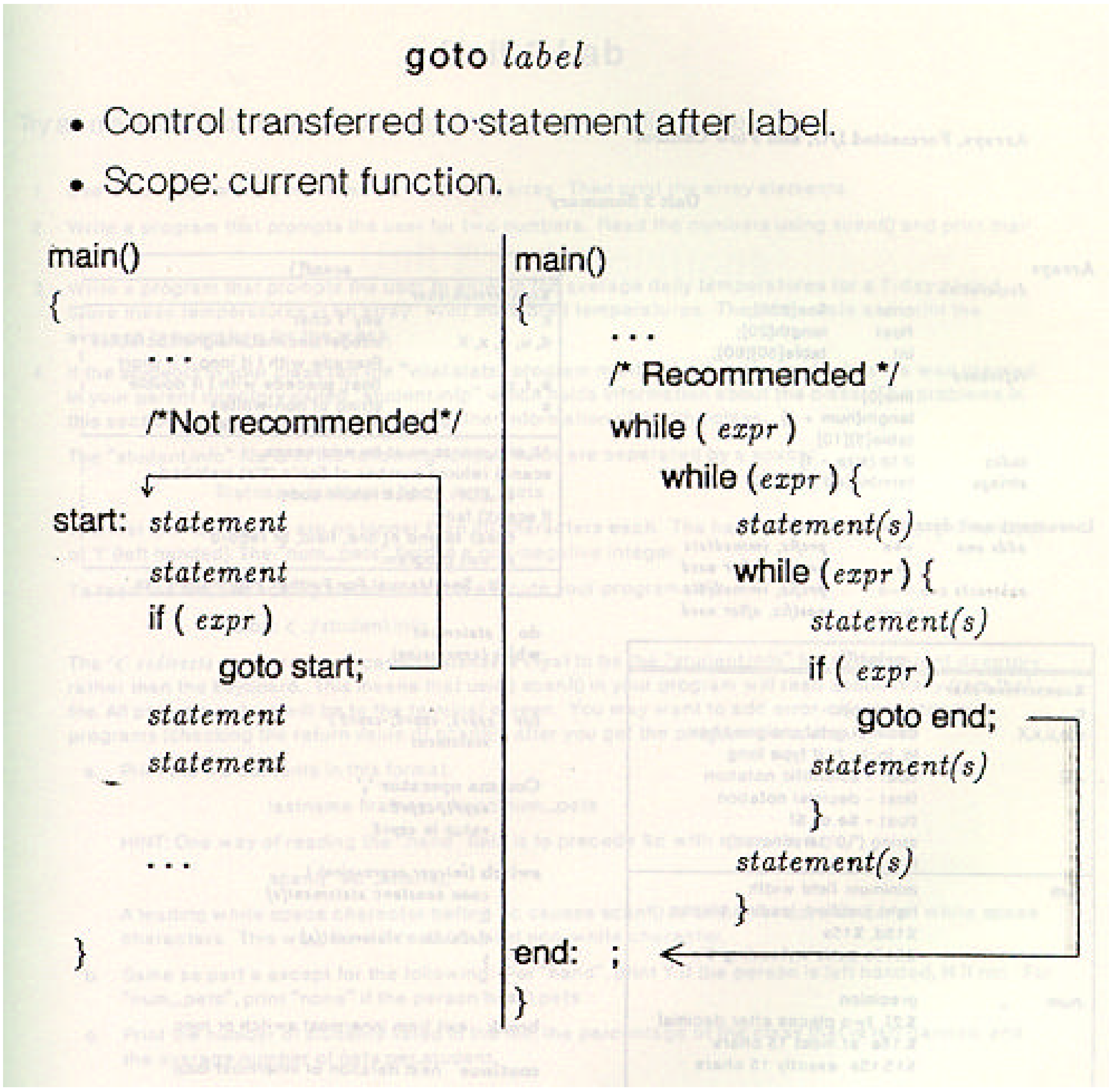
```
switch (expression) {
        case constant:    statement(s)
        case constant:    statement(s)
        default:          statement(s)
        }
```

```
switch (num)  {
   case 1:  statement;
            break;
   case 10:  statement;
            statement;
            break;
   case 100: statement;
            break;
   default:  printf("Error\n");
            break;
   }
```

```
if (num == 1)
    statement;
else if (num == 10) {
    statement;
    statement;
    }
else if (num == 100)
    statement;
else
    printf("Error\n");
```

switch statement - fall through

```
input = getchar();
switch(input) {
      case 'a':
      case 'A': statement    /* Add record */
                statement
                break;
      case 'd':
      case 'D': statement    /* Delete record */
                statement
                break;
      default:  printf("Illegal choice\n");
                break;
      }
```



goto *label*

- Control transferred to statement after label.
- Scope: current function.

```
main()
{
      . . .
      /*Not recommended*/

start: statement
       statement
       if ( expr )
              goto start;
       statement
       statement

       . . .

}
```

```
main()
{
      . . .
      /* Recommended */
      while ( expr )
          while (expr ) {
                 statement(s)
                 while (expr ) {
                        statement(s)
                        if ( expr )
                              goto end;
                        statement(s)
                 }
                 statement(s)
          }
end:  ;
}
```