Chapter 9

Programming Languages

OBJECTIVES

After reading this chapter, the reader should be able to:

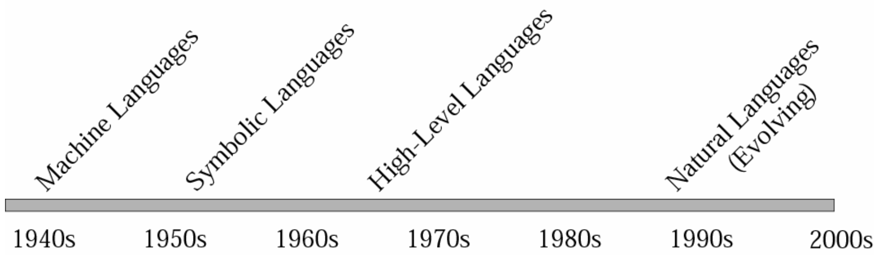
- Have a vision of computer language evolution.
- Distinguish between machine, assembly, and high-level languages.
- Understand the process of creating and running a program.
- Distinguish between the different categories of languages: procedural, object-oriented, functional, declarative, and special.
- Become familiar with elements of the procedural language C.

9.1

EVOLUTION

Figure 9-1

Evolution of computer languages



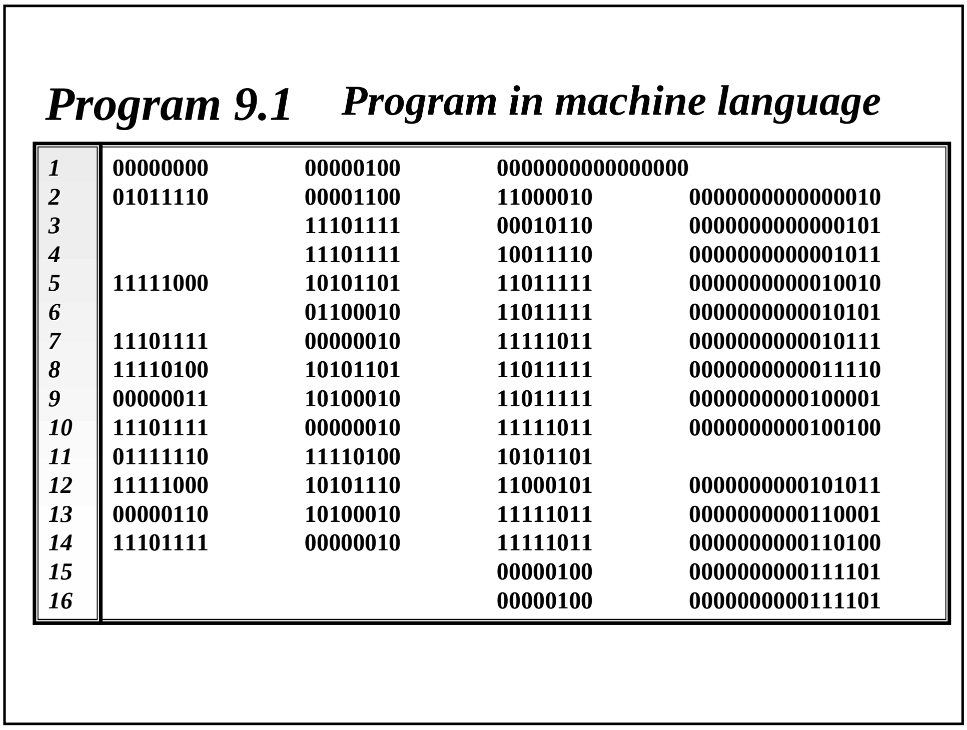
Machine Languages

Symbolic Languages

High-Level Languages

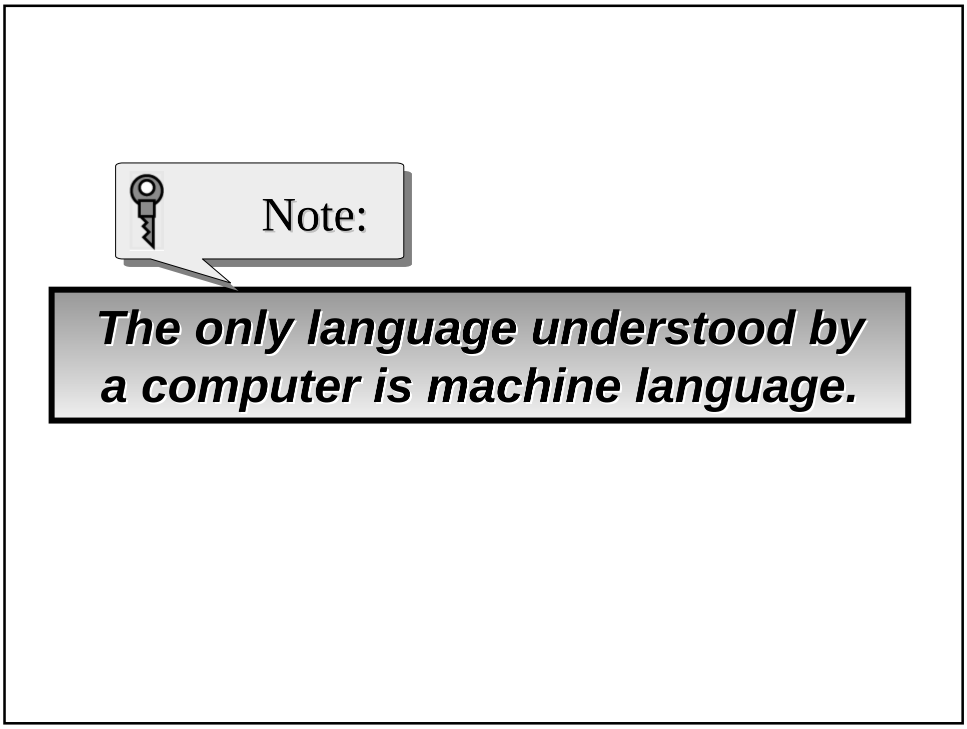
Natural Languages (Evolving)

1940s 1950s 1960s 1970s 1980s 1990s 2000s



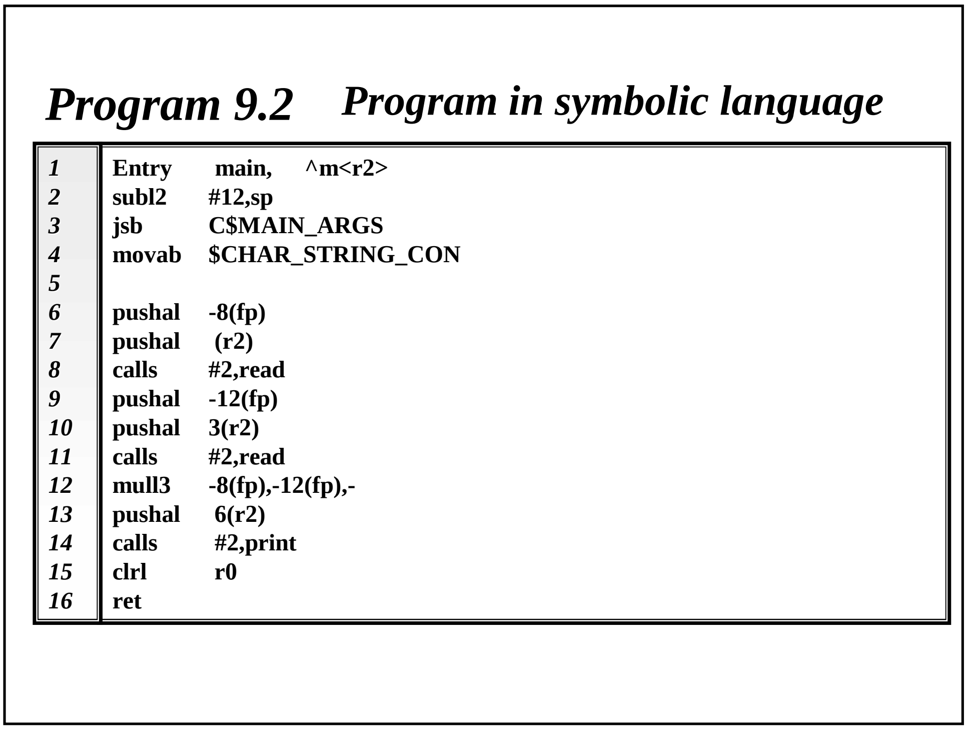
Program 9.1 *Program in machine language*

```
00000000    00000100    0000000000000000
01011110    00001100    11000010    0000000000000010
            11101111    00010110    0000000000000101
            11101111    10011110    0000000000001011
11111000    10101101    11011111    0000000000010010
            01100010    11011111    0000000000010101
11101111    00000010    11111011    0000000000010111
11110100    10101101    11011111    0000000000011110
00000011    10100010    11011111    0000000000100001
11101111    00000010    11111011    0000000000100100
01111110    11110100    10101101
11111000    10101110    11000101    0000000000101011
00000110    10100010    11111011    0000000000110001
11101111    00000010    11111011    0000000000110100
                        00000100    0000000000111101
                        00000100    0000000000111101
```



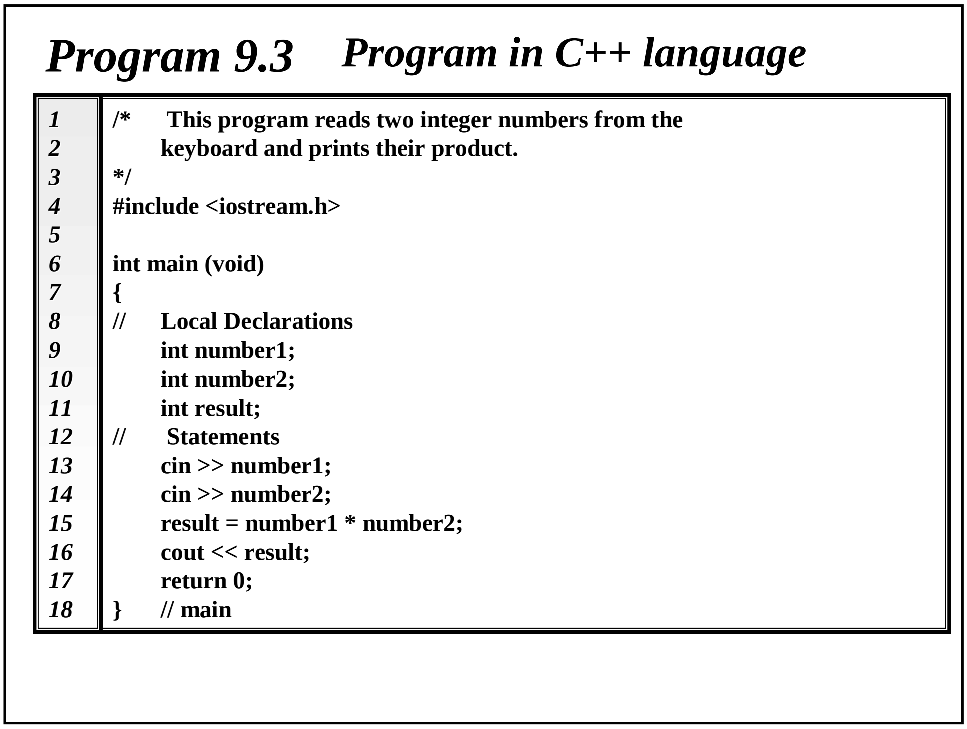
Note:

The only language understood by a computer is machine language.



Program 9.2 *Program in symbolic language*

```
Entry      main,     ^m<r2>
subl2      #12,sp
jsb        C$MAIN_ARGS
movab      $CHAR_STRING_CON

pushal     -8(fp)
pushal      (r2)
calls      #2,read
pushal     -12(fp)
pushal     3(r2)
calls      #2,read
mull3      -8(fp),-12(fp),-
pushal      6(r2)
calls       #2,print
clrl        r0
ret
```



Program 9.3 *Program in C++ language*

```
/*      This program reads two integer numbers from the
        keyboard and prints their product.
*/
#include <iostream.h>

int main (void)
{
//      Local Declarations
        int number1;
        int number2;
        int result;
//       Statements
        cin >> number1;
        cin >> number2;
        result = number1 * number2;
        cout << result;
        return 0;
}       // main
```

Natural Language

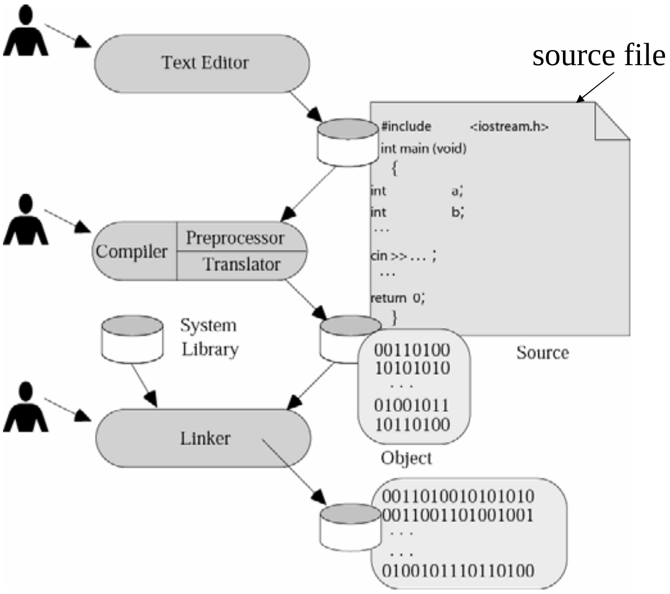
- Natural language
  - – English, Chinese
- Artificial language
  - – BASIC, C, Java
- Can computer understand natural language?
- Natural language is “context-sensitive”!
  - – 打人
  - – 打電話
  - – 打開

9.2

BUILDING A PROGRAM

Figure 9-2

Building a program



```
#include        <iostream.h>
int main (void)
  {
int             a;
int             b;
 . . .
cin >> . . . ;
 . . .
return 0;
  }
```

```
00110100
10101010
 . . .
01001011
10110100
```

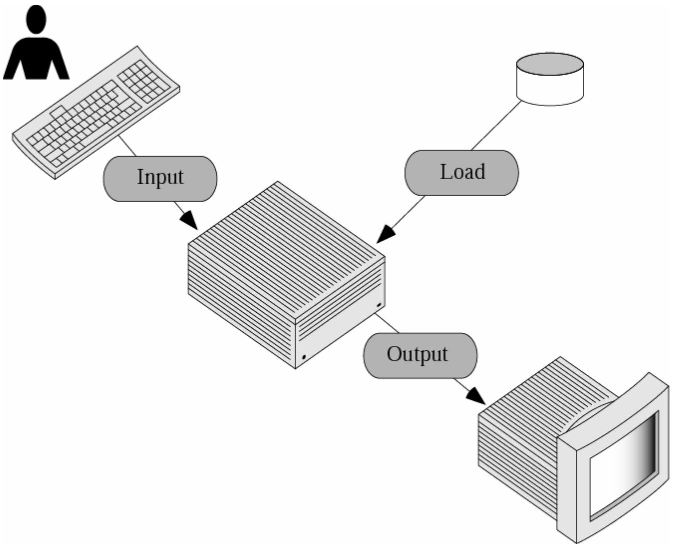
```
0011010010101010
0011001101001001
 . . .
 . . .
0100101110110100
```

9.3

PROGRAM EXECUTION

Figure 9-3

Program execution

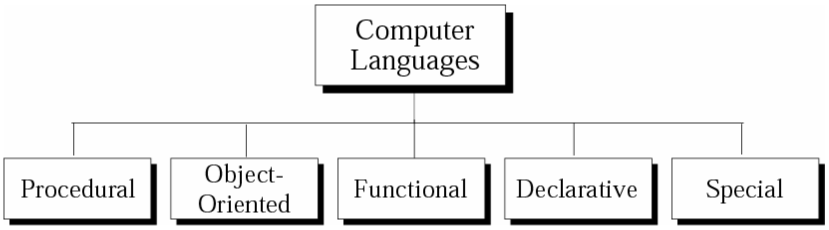


9.4

CATEGORIES OF LANGUAGES

Figure 9-4

Categories of languages



Procedural Language (Imperative Language)

- A set of instructions that are executed one by one.
  - – Some may be skipped => Decision
  - – Some may be executed more than once=> Repetition.
- Example
  - – FORTRAN (Formula Translation)
    - The first high level language.
    - Jack Backus, 1957
    - Scientific and engineering computation.
      - – High precision, complex computation

Procedural Language

- COBOL (Common Business-Oriented Language)
  - Grace Hopper of the U.S. navy.
  - Business programming language.
    - Precise computation is not important.
    - Fast access /update the files and the database.
- Pascal
  - Niklaus 1971 (Named after Pascal— French mathematician)
  - structured programming approach
  - Popular in academia

Procedural Language

- C
  - Dennis Ritche 1970
  - Designed for writing OS and system software
  - Popular today
    - High level and low level features.
    - Standardized by ANSI and ISO
- Ada
  - Created for U.S. department of defense
  - Real-time processing
  - parallel processing

Object-oriented language

- Object can be viewed as a data item.
- In procedural language: objects are separated from operations. (Objects are passive).
- In OOP, objects are active.
-

```
class complex                      complex a,b;
{                    ———→          a.real=1;
  double real, image;              …
  add(complex c);                  a.add(b);
  sub(complex c);                  b.sub(a);
}
```

Object-oriented language

- Example:
  - C++: Bjarne Stroustrup at Bell Lab.
    - Encapsulation:
      - Operate the data through the interface.
    - Inheritance:
      - Geometric_shape=> area, mass center
      - Triangle=> Geometric_shape +length, height
      - Circle=> Geometric_shape +radius
    - Polymorphism:
      - Define several operations with the same names.

A Simple Example

Interface:

```
class Geometric_Shape
{
  double area,  x, y;
  double SetCenter(x,y);
}
class Triangle::public Geometric_Shpae
{
  double length, height;
  double ComputeArea();
  double ComputeArea(e1, e2, angle);
}
```

Main program:

```
Triangle a;
a.SetCenter(10,10);
a.ComputeArea(10,10,0.1);
a.ComputeArea();
```

Object-oriented language

– Java:
  - Developed by Sun, based on C++
  - No multiple inheritance, pointers,
  - Pure OOP.
  - Stand-alone program and applet
  - Run through Java virtual machine.
  - multithreading

Figure 9-5

Function in a functional language

A program is considered a mathematical function.
Combine functions to create a new one.

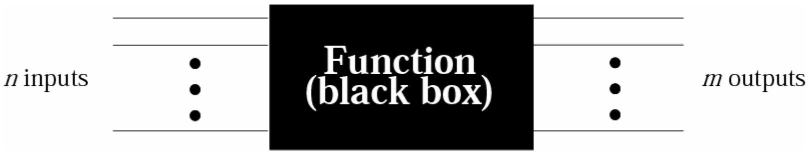
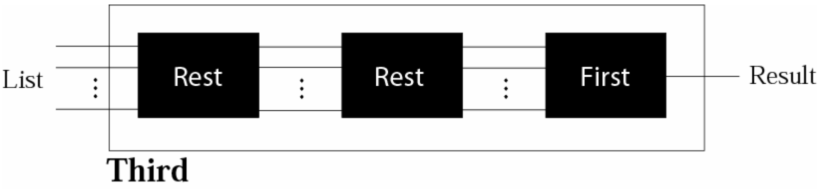
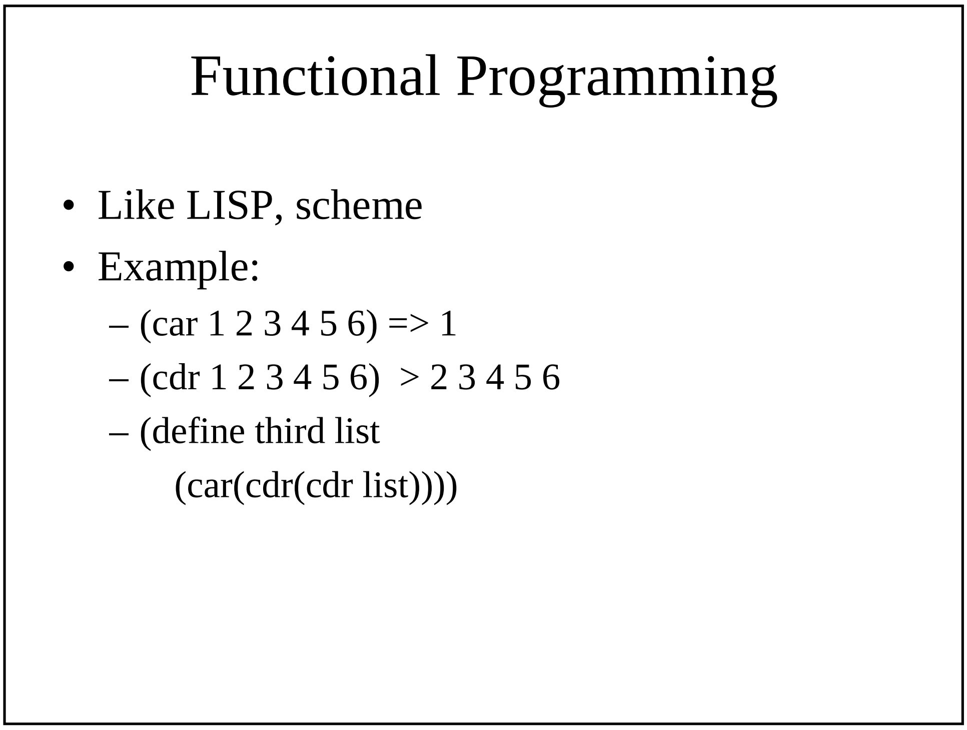


Figure 9-6

Extracting the third element of a list

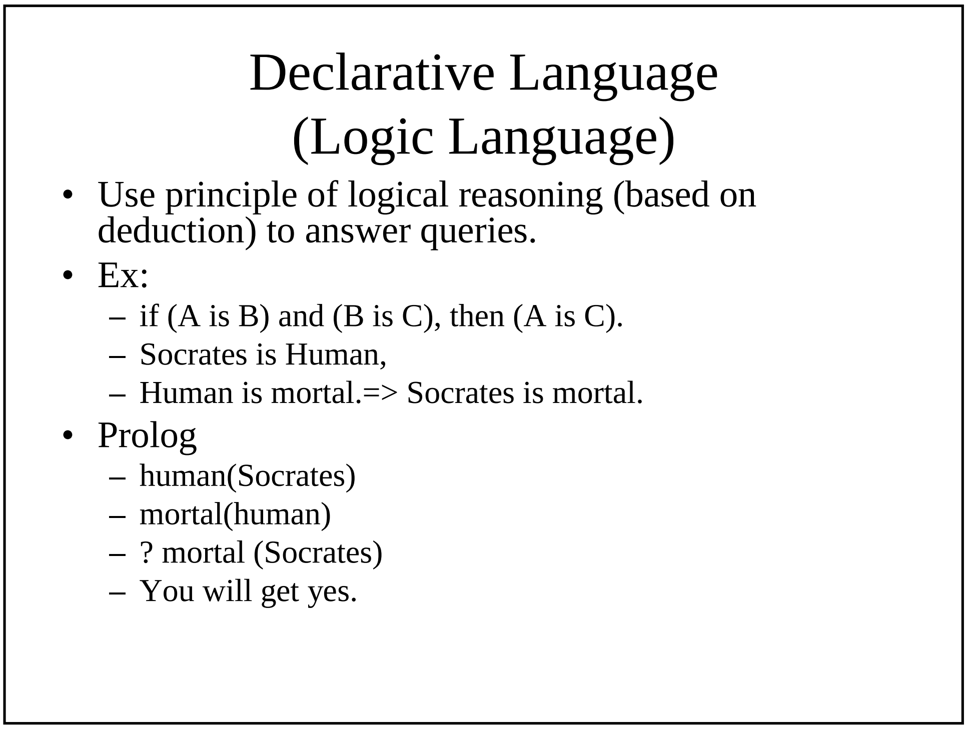




Functional Programming

- Like LISP, scheme
- Example:
  - (car 1 2 3 4 5 6) => 1
  - (cdr 1 2 3 4 5 6) > 2 3 4 5 6
  - (define third list

    (car(cdr(cdr list))))



Declarative Language (Logic Language)

- Use principle of logical reasoning (based on deduction) to answer queries.
- Ex:
  - if (A is B) and (B is C), then (A is C).
  - Socrates is Human,
  - Human is mortal.=> Socrates is mortal.
- Prolog
  - human(Socrates)
  - mortal(human)
  - ? mortal (Socrates)
  - You will get yes.

Other Special Languages

- HTML (Hypertext markup language)
  - – Use ASCII for the text and the instructions.
  - – Write webpages.
  - – HTML can be divided into
    - head: title or information for the browser
    - body: content
    - tags: instructions.

Table 9.1 Common tags

Beginning Tag	***Ending Tag***	***Meaning***
----------------	**----------------**	[redacted]
<HTML>	**</HTML>**	[redacted]
<HEAD>	**</HEAD>**	[redacted]
<BODY>	**</BODY>**	[redacted]
<TITLE>	**</TITLE>**	[redacted]
<Hi>	**</Hi>**	[redacted]
<B>	**</B>**	[redacted]
<I>	**</I>**	[redacted]
<U>	**</U>**	[redacted]
<SUB>	**</SUB>**	[redacted]
<SUP>	**</SUP>**	[redacted]
<CENTER>	**</CENTER>**	[redacted]
<BR>		[redacted]
<OL>	**</OL>**	[redacted]
<UL>	**</UL>**	[redacted]
<LI>	**</LI>**	[redacted]
<IMG>		[redacted]
<A>	**</A>**	[redacted]

Program 9.4 HTML Program

```
<HTML>
        <HEAD>
                <TITLE> Sample Document </TITLE>
        </HEAD>
        <BODY>
                This is the  <H1> Largest </H1> <H2>Larger</h2>
                <U> Underline</U> letter.
        </BODY>
</HTML>
```

Other Special Languages

- Perl:
  - Used to parse a string and extract information.
- SQL:
  - A language used to interact with database.

9.5

A PROCEDURAL LANGUAGE: C

Homework

- Read this section yourself.
- Explain the following ten terms in next class.

symbolic constant, variable declaration, compound statements, logical operators, pass by value, pass by reference, function declaration, switch statement, for loop, do-while loop.

Figure 9-7

Variables

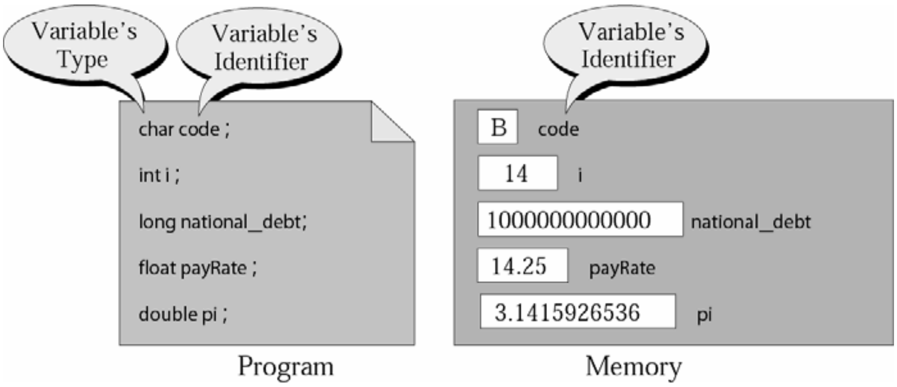


Table 9.2 Arithmetic operators

Operator	*Definition*	*Example*
+	**Addition**	[redacted]
−	**Subtraction**	[redacted]
*	**Multiplication**	[redacted]
/	**Division (quotient)**	[redacted]
%	**Division (remainder)**	[redacted]
++	**Increment**	[redacted]
––	**Decrement**	[redacted]

Table 9.3 Relational operators

Operator	*Definition*	*Example*
----------------	----------------	[redacted]
<	**Less than**	
<=	**Less than or equal to**	
>	**Greater than**	
>=	**Greater than or equal to**	
==	**Equal to**	
!=	**Not equal to**	

Table 9.4   Logical operators

Operator	*Definition*	*Example*
----------------	----------------	[redacted]
!	**NOT**	
&&	**AND**	
||	**OR**	

Table 9.5 Assignment operators

Operator	*Example*
----------------	----------------
==	Num = 5
+=	Num += 5
–=	Num –= 5
*=	Num *= 5
/=	Num /= 5
%=	Num %= 5

[redacted]

Figure 9-8

Statements

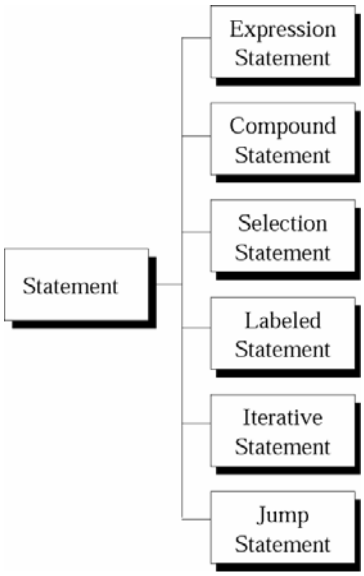


Figure 9-9

Side effect of a function

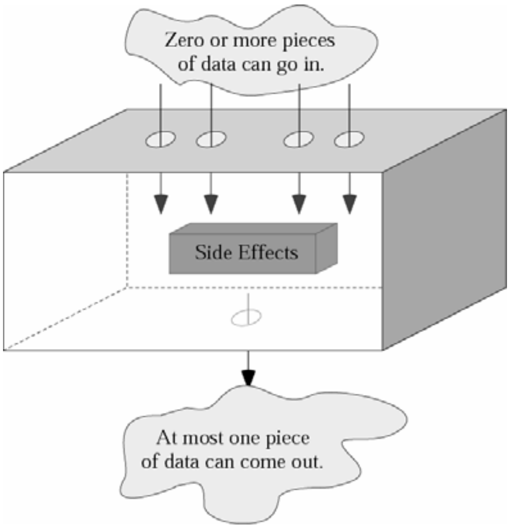
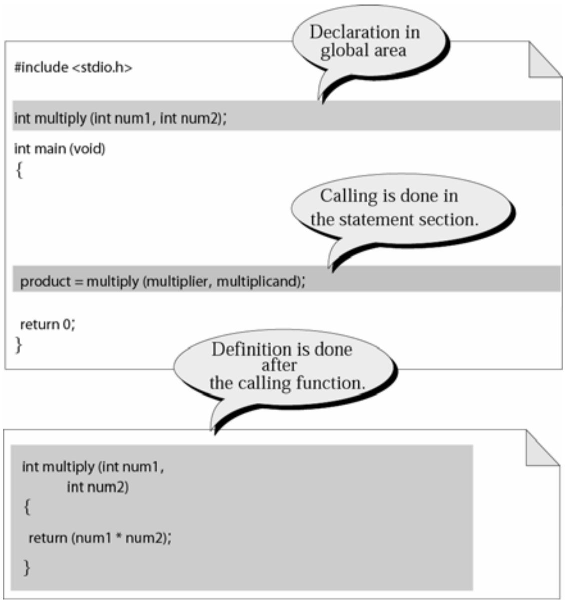

Zero or more pieces of data can go in.
Side Effects
At most one piece of data can come out.

Figure 9-10

Function declaration

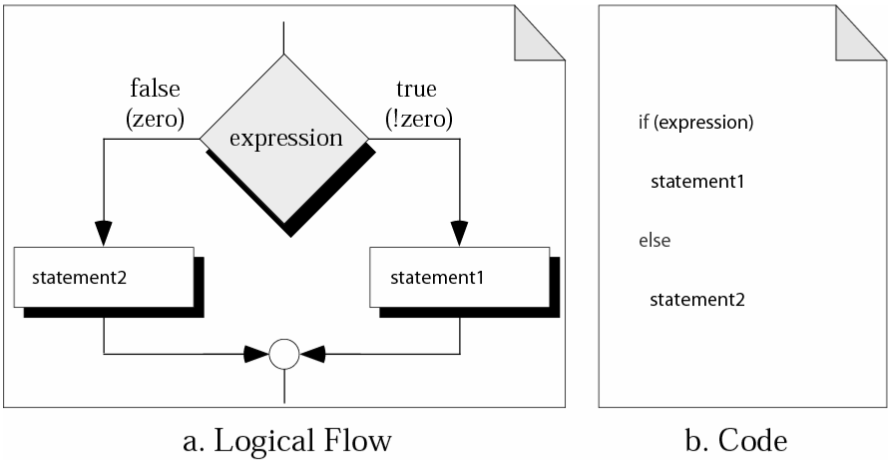

Declaration in global area
Calling is done in the statement section.
Definition is done after the calling function.

```
#include <stdio.h>

int multiply (int num1, int num2);
int main (void)
{

 product = multiply (multiplier, multiplicand);

 return 0;
}
```

```
int multiply (int num1,
          int num2)
{
 return (num1 * num2);
}
```

Figure 9-11

if-else statement



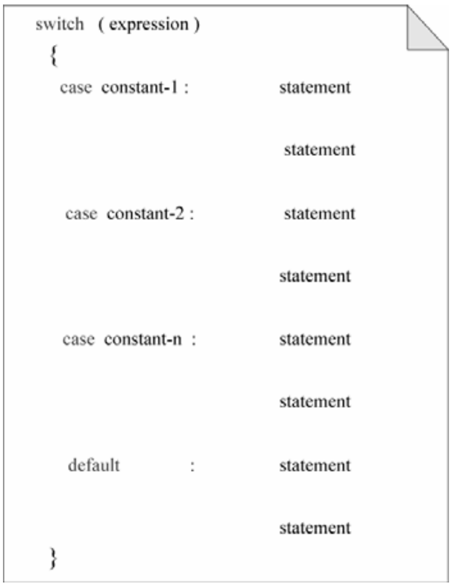
```
if (expression)
  statement1
else
  statement2
```

a. Logical Flow

b. Code

Figure 9-12

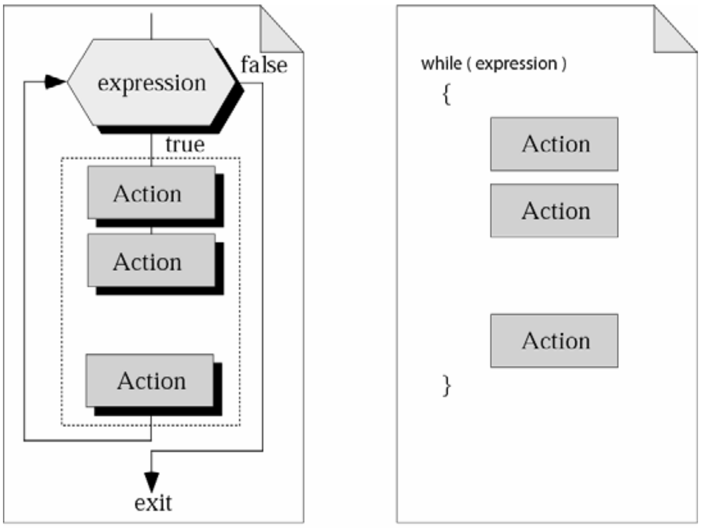
switch statement



```
switch  ( expression )
 {
   case  constant-1 :          statement
                                statement
    case  constant-2 :          statement
                               statement
   case  constant-n  :         statement
                               statement
    default          :         statement
                               statement
 }
```

Figure 9-13

while loop



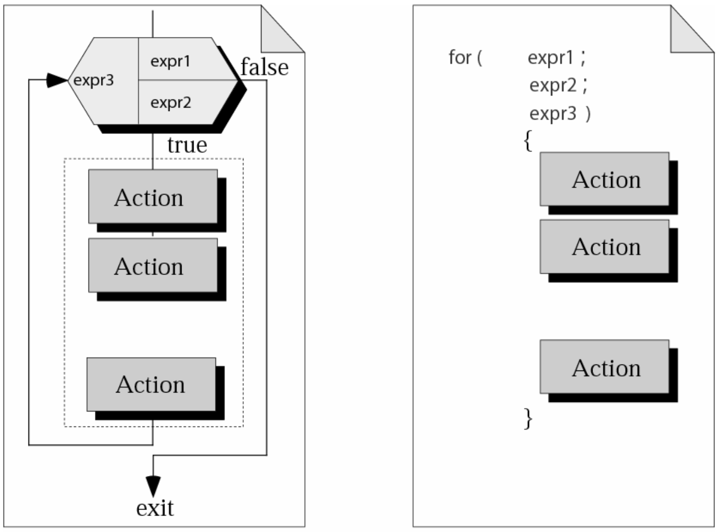
```
while ( expression )
  {
      Action
      Action

      Action
  }
```

a. Flowchart

b. C Language

Figure 9-14

for loop



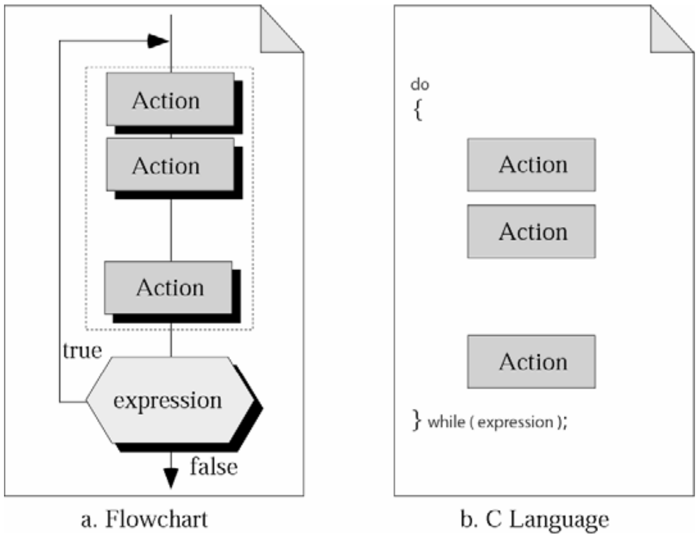
```
for (     expr1 ;
          expr2 ;
          expr3 )
          {
              Action
              Action

              Action
          }
```

a. Flowchart

b. C Language

Figure 9-15

do-while loop



a. Flowchart

b. C Language