An Example

Compute Squares

- Prints a table of squares as shown below

1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

```
/* Compute Squares */
#include <stdio.h>

main() {
  int    n, n_square;
  int    lower, upper, step;

  lower = 1;      /* Lower imit */
  upper = 10;     /* Upper limit */
  step = 1;

  n = lower;
  while ( n <= upper ) {
    n_square = n * n;
    printf("%3d\t%6d\n",n,n_square);
    n = n + 1;
  }
}
```

An Example

Functions

- C programs contain functions and variables
- Function names are followed by a list of arguments enclosed in parenthesis.
- Function definition consists of a group of statements enclosed in curly brackets.
- **`printf`** is a library function. The information about this function is contained in a file named stdio.h.

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Variables

- Variables refer the objects that can be stored in computer's memory locations. Variables are named (identifiers)
- Variables can store data of various types. Some basic datatypes are

`char`	characters (1/2 bytes)
`int`	integers (2/4 bytes)
`float`	rationals (4 bytes)
`double`	rationals(8 bytes)

- Variables must be declared and initialized.

```
/* Compute Squares */
#include <stdio.h>

main() {
  int    n, n_square;
  int    lower, upper, step;

  lower = 1;      /* Lower imit */
  upper = 10;     /* Upper limit */
  step = 1;

  n = lower;
  while ( n <= upper ) {
    n_square = n * n;
    printf("%3d\t%6d\n",n,n_square);
    n = n + 1;
  }
}
```

An Example

Declarations

- Declarations have form

```
type var1, var2, var3;
```

- Merely announces the data type to be associated with a variable.
- Initial value of variable is not known.
- Examples

```
char name;
int numStudents;
float applePrice;
double veryAccurateVar;
```

```
/* Compute Squares */
#include <stdio.h>

main() {
  int   n, n_square;
  int   lower, upper, step;

  lower = 1;      /* Lower imit */
  upper = 10;     /* Upper limit */
  step = 1;

  n = lower;
  while ( n <= upper ) {
    n_square = n * n;
    printf("%3d\t%6d\n",n,n_square);
    n = n + 1;
  }
}
```

An Example

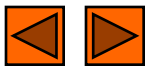
- Assignments Statement
  - The assignment statements have the following form

    ```
    someVar = expression ;
    ```

  - Expressions consist of variables, functions and operators.
  - Examples

    ```
    c = 5 * ( f - 32 )  /  9
    s = sin(x) / x
    y = log(x) + v0
    I = P * R * T / 100
    Tax = 0.3 * ( I - 60000 )
    ```

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```



An Example

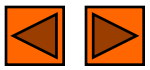
Loops

- The form of the while statement is

```
While ( condition )
   statement ;
While ( condition ) {
   statements }
```

- If the condition is true the statements in the body of the while loop are executed. At the end of the loop the the condition is tested again and executes the loop if the condition is true. This procedure continues till the condition fails to be true.
- Conditions are usually logical expressions using operators like ==, <, <=, >, >= etc. These have boolean value

```
/* Compute Squares */
#include <stdio.h>

main() {
  int    n, n_square;
  int    lower, upper, step;

  lower = 1;      /* Lower imit */
  upper = 10;     /* Upper limit */
  step = 1;

  n = lower;
  while ( n <= upper ) {
    n_square = n * n;
    printf("%3d\t%6d\n",n,n_square);
    n = n + 1;
  }
}
```



An Example

- Execution

Variable	Before exeution	After execution
n	??	??
n_square	??	??
lower	??	??
upper	??	??
step	??	??

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

- Execution

Variable	Before exeution	After execution
n	??	??
n_square	??	??
lower	??	??
upper	??	??
step	??	??

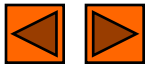
```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

Variable	Before exeution	After execution
n	??	??
n_square	??	??
lower	??	1
upper	??	??
step	??	??

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```



An Example

Execution

Variable	Before exeution	After execution
n	??	??
n_square	??	??
lower	1	1
upper	??	10
step	??	??

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

- Execution

Variable	Before exeution	After execution
n	??	??
n_square	??	??
lower	1	1
upper	10	10
step	??	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

- Execution

Variable	Before exeution	After execution
n	??	1
n_square	??	??
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	1	1
n_square	??	??
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	1	1
n_square	??	1
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	1	1
n_square	1	1
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	1	2
n_square	1	1
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
  int   n, n_square;
  int   lower, upper, step;

  lower = 1;      /* Lower imit */
  upper = 10;     /* Upper limit */
  step = 1;

  n = lower;
  while ( n <= upper ) {
    n_square = n * n;
    printf("%3d\t%6d\n",n,n_square);
    n = n + 1;
  }
}
```

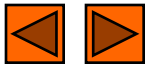
An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	2	2
n_square	1	1
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```



An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	2	2
n_square	1	4
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

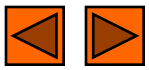
An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	2	2
n_square	4	4
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```



An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	2	3
n_square	4	4
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is true.

Variable	Before exeution	After execution
n	10	11
n_square	100	100
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is false.

Variable	Before exeution	After execution
n	11	11
n_square	100	100
lower	1	1
upper	10	10
step	1	1

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

An Example

Execution

- `n <= upper` is false.

Variable	Before exeution	After execution
n	11	11
n_square	100	100
lower	1	1
upper	10	10
step	1	1

- **Program ends**

```
/* Compute Squares */
#include <stdio.h>

main() {
   int    n, n_square;
   int    lower, upper, step;

   lower = 1;      /* Lower imit */
   upper = 10;     /* Upper limit */
   step = 1;

   n = lower;
   while ( n <= upper ) {
     n_square = n * n;
     printf("%3d\t%6d\n",n,n_square);
     n = n + 1;
   }
}
```

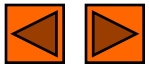
Printf Function

- The first argument of printf is a *format specifier.*
- Each % sign in the format specifier is a formatting instruction.
- `%d` print a decimal integer
- `%f` print a floating number
- `%wd` prints a decimal number with min width of w chars
- `%w.pf` prints a floating number with precision p and min width w
- Other characters in the format specifier are just printed.

Printf Function

- Examples

```
Printf( "I am Charu" );                    I am Charu
Printf( "|%3d|" , 8 );                     |  8|
Printf( "|%03d|" , 8 );                    |8  |
Printf( "|%3d|" , 8000 );                  |8000|
Printf( "|%f|" , 123.456 );                |123.456000|
Printf( "|%f|" , 123.4567890123 );         |123.456789|
Printf( "|%8.3d|" , 123.4567890123 );      | 123.456|
Printf( " Age = %d years" , 22 );          Age = 22 years
Printf( "%d and %d" , 1, 2 );                    1 and 2
```



An Example

Area of a circle

- The program produces the following output when 5.0 is input

```
Enter the radius 5.0
Area = 78.539749
Peri = 31.415899
```

- And produces the following when –3.0 is input.

```
Enter the radius -3.0
Negative radius
```

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

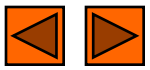
Decisions

- The form of **if-else** statement is as follows:

```
if ( condition ) statements

if ( condtion ) statements
else statements
```

- If condition is true then **if-block** of statements is executed otherwise **else-block** of statements is executed.

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```



An Example

⌘Execution

Variable	Before exeution	After execution
rad	??	??
area	??	??
peri	??	??

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

Variable	Before exeution	After execution
rad	??	5.000000
area	??	??
peri	??	??

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

- Condition `rad > 0.0` is true

Variable	Before exeution	After execution
rad	5.000000	5.000000
area	??	??
peri	??	??

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

Variable	Before exeution	After execution
rad	5.00000	5.00000
area	??	78.53975
peri	??	??

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

Variable	Before exeution	After execution
rad	5.00000	5.00000
area	78.53975	78.53975
peri	??	31.41590

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

⌘Execution

Variable	Before exeution	After execution
rad	5.00000	5.00000
area	78.53975	78.53975
peri	31.41590	31.41590

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

➡   printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

Variable	Before exeution	After execution
rad	5.00000	5.00000
area	78.53975	78.53975
peri	31.41590	31.41590

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

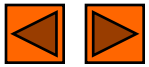
An Example

Execution

Variable	Before exeution	After execution
rad	5.00000	5.00000
area	78.53975	78.53975
peri	31.41590	31.41590

Program ends

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```



An Example

Execution

Variable	Before exeution	After execution
rad	??	-3.00000
area	??	??
peri	??	??

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

- The condition `rad > 0.0` is false

Variable	Before exeution	After execution
rad	-3.00000	-3.00000
area	??	??
peri	??	??

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

An Example

Execution

- The condition `rad > 0.0` is false

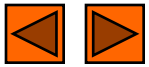
Variable	Before exeution	After execution
rad	-3.00000	-3.00000
area	??	??
peri	??	??

- Prints "**Negative radius**" and
- Program ends

```
/* Compute Area and Perimeter of  a
   circle */
#include <stdio.h>

main() {
  float  rad;
  float area, peri;

  printf( "Enter the radius " );
  scanf("%f" , &rad);

  if ( rad > 0.0 ) {
    area = 3.14159 * rad * rad;
    peri = 6.28318 * rad;

    printf( "Area = %f\n" , area );
    printf( "Peri = %f\n" , peri );
  }
  else
    printf( "Negative radius\n");
}
```

Scanf Function

- The first argument of scanf is a *format specifier.*
- Each % sign in the format specifier is a formatting instruction.
- **`%d`** scans a decimal integer and stores it in **`int`** variable
- **`%f`** scans a floating number and stores it in **`float`** variable
- **`%wd`** maximum width of w chars are scanned
- **`%wf`** maximum width of w chars are scanned
- Other characters in the format specifier are expected to be input as is. Best to avoid any other chars in the input.



Scanf Function

scanf (format, &intvar, &fltvar);

Format	"%d%f"		"%3d%6f"	
Input	Intvar	Fltvar	Intvar	Fltvar
3 4.5	3	4.5	3	4.5
3\t4.5 (tab)	3	4.5	3	4.5
3\n4.5(newline)	3	4.5	3	4.5
34.5	34	0.5	34	0.5
003 4.5	3	4.5	3	4.5
0034.5	34	0.5	3	4.5
0003 4.5	3	4.5	0	3.0
00034.5	34	0.5	0	34.5
3 000004.5	3	4.5	3	4.0
3 4.5678912	3	4.567891	3	4.567800
A3a4.5	??	??	??	??
3a4.5	3	??	3	??

Algorithms

Problem

- Let S be a finite sequence of positive nonzero integers a_k, except the last number which is always 0. Find the largest number in the sequence.
- Example
- 7, 9, 4, 6, 2, 5, 8, 0
- The largest is 9.
- If S_k is a subsequence of first k numbers of S and m_k is the largest term of S_k then
- $m_{k+1} = \max \{ m_k , a_{k+1} \}$

Algorithm

1. The m_1 be the largest in S_1. Clearly $m_1 = a_1$.
2. Let I = 1.
3. If a_{I+1} is zero, go to step 7.
4. $m_{I+1} = \max \{ m_I, a_{I+1} \}$.
5. Increment I.
6. Go to step 3.
7. Print m_I.

An Example

Find Largest

- The program produces the following output

```
Enter a number 7
Enter a number 9
Enter a number 4
Enter a number 6
Enter a number 2
Enter a number 0
Largest is 9
```

```
/* Find the largest number */
#include <stdio.h>

main() {
  int number, largest;

  printf( "Enter a number " );
  scanf( "%d" , &largest );

  printf( "Enter a number " );
  scanf( "%d" , &number );

  while ( number > 0 ) {
    if ( number > largest )
      largest = number;
    printf( "Enter a number " );
    scanf( "%d" , &number );
  }

  printf( "Largest is %d\n" ,
          largest );
}
```