CENG 230

Introduction to C Programming

Week 5 – Selection

Sinan Kalkan

Some slides/content are borrowed from Tansel Dokeroglu, Nihan Kesim Cicekli.

Previously on CEng230!

Logical Operators

0 <= n && n <= 100

!(0 <= n && n <= 100)

- && || !

Operator	Type	Associativity
+ - ++ -- !	Unary	Right to left
* / %	Binary	Left to right
+ -	Binary	Left to right
< <= > >=	Binary	Left to right
== !=	Binary	Left to right
&&	Binary	Left to right
||	Binary	Left to right
= *= /= %= += -=	Binary	Right to left

TABLE 4.3 The && Operator (and)

operand1	operand2	operand1 && operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	0 (false)
0 (false)	nonzero (true)	0 (false)
0 (false)	0 (false)	0 (false)

TABLE 4.4 The || Operator (or)

operand1	operand2	operand1 || operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	1 (true)
0 (false)	nonzero (true)	1 (true)
0 (false)	0 (false)	0 (false)

TABLE 4.5 The ! Operator (not)

operand1	!operand1
nonzero (true)	0 (false)
0 (false)	1 (true)

C accepts *any nonzero value as a representation of true.*

short-circuit evaluation

An expression of the form (a || b)
must be true if a is true. Consequently, C stops evaluating the expression when it determines that the value of !flag is 1 (true).

Similarly, an expression of the form (a && b)
must be false if a is false, so C would stop evaluating such an expression if its first operand evaluates to 0.

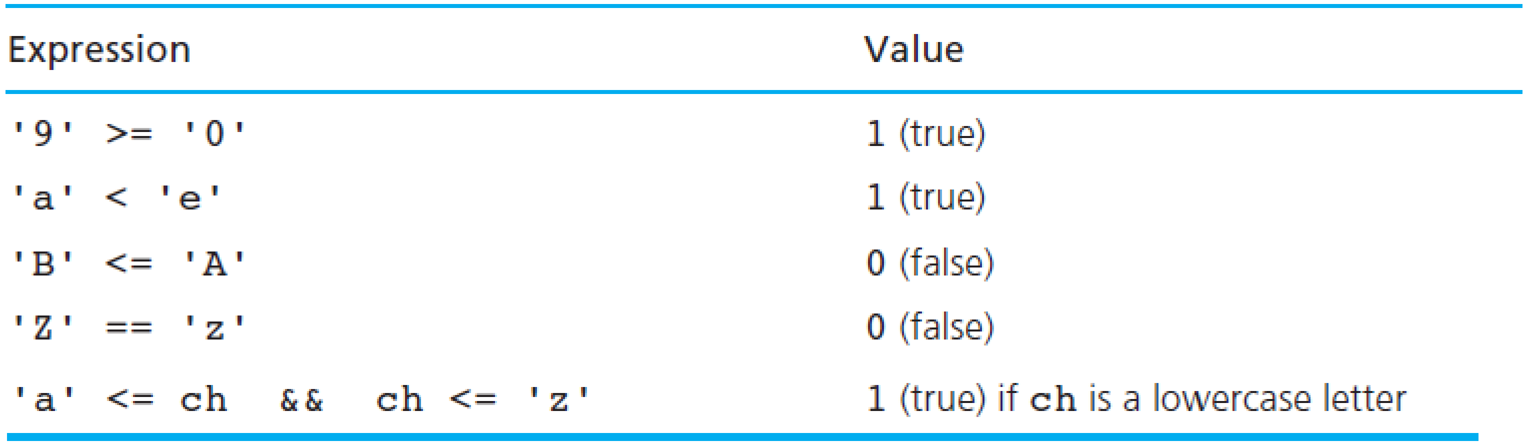
Writing English Conditions in C

x is 3.0
y is 4.0
z is 2.0

English Condition	Logical Expression	Evaluation
`x` and `y` are greater than `z`	`x > z && y > z`	`1 && 1` is 1 (true)
`x` is equal to `1.0` or `3.0`	`x == 1.0 || x == 3.0`	`0 || 1` is 1 (true)
`x` is in the range `z` to `y`, inclusive	`z <= x && x <= y`	`1 && 1` is 1 (true)

`x` is outside the range `z` to `y`	`!(z <= x && x <= y)`
	`z > x || x > y`
		`!(1 && 1)` is 0 (false)
		`0 || 0` is 0 (false)

Comparing Characters



Expression	Value
`'9' >= '0'`	1 (true)
`'a' < 'e'`	1 (true)
`'B' <= 'A'`	0 (false)
`'Z' == 'z'`	0 (false)
`'a' <= ch  &&  ch <= 'z'`	1 (true) if **ch** is a lowercase letter

Examples

int a = 6 , b = 9 , c = 14 , flag = 1 .

```
c == a + b  ||  !flag
a != 7  &&  flag  ||  c >= 6
!(b <= 12)  &&  a % 2 == 0
!(a > 5  ||  c < a + b)
```

int ans;
int p = 100, q = 50.

ans = (p > 95) + (q < 95);
What is the value of ans?

Complement the expression below

a != 7 && flag || c >= 6

a == 7 || flag && c < 6

!(1 || 0)	0
!(1 || 1 && 0)	0
!((1 || 0) && 0)	1 (Parenthesis are useful)

Previously on CEng 230!

Type conversion (casting)

Type conversions (casting)

Previously on CEng 230!

```
float a = 5.25;
int b = a;
/*Casting from float to int. The value of b here is 5*/


char c = ‘A’;
int x = c;
/*Casting from char to int.
The value of x here is 65: the ASCII code of ‘A’*/


int x=7, y=5 ;
float z;
z=x/y;
/* the value of z is 1.00 */


int x=7, y=5;
float z;
z = (float)x/(float)y;
/ the value of z is 1.4*/
```

Type conversions (casting)

Previously on CEng 230!

```
printf( "Welcome : %d", (3/2) );
```

Output is : 1 and **fraction** part of the number is lost

```
int sum = 17, count = 5;
double mean;
mean = (double) sum / count;
printf("Value of mean : %f\n", mean );
```

```
Value of mean : 3.400000
```

```
int i = 17;
char c = 'c'; /* ascii value is 99 */
int sum;
sum = i + c;
printf("Value of sum : %d\n", sum );
```

```
Value of sum : 116
```

Previously on CEng 230!

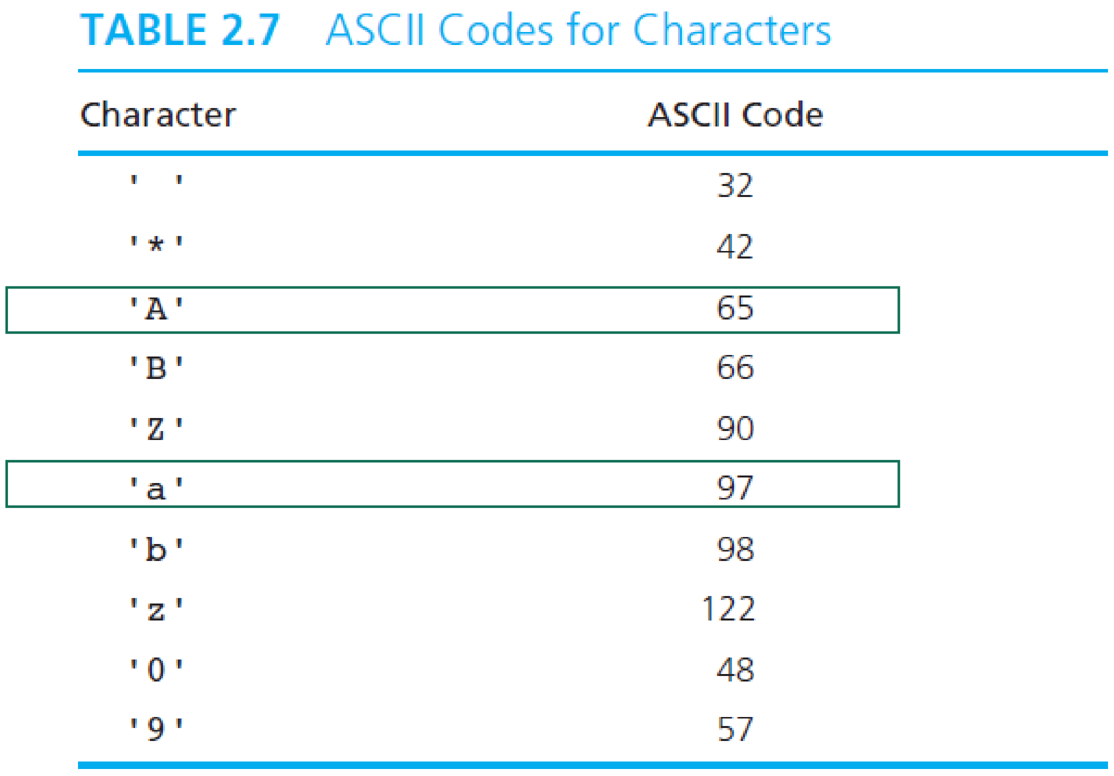


TABLE 2.7 ASCII Codes for Characters

Character	ASCII Code
`' '`	32
`'*'`	42
`'A'`	65
`'B'`	66
`'Z'`	90
`'a'`	97
`'b'`	98
`'z'`	122
`'0'`	48
`'9'`	57

What is the result of printf("%d", ‘d’– ‘a’);

Type Conversion

Previously on CEng230!

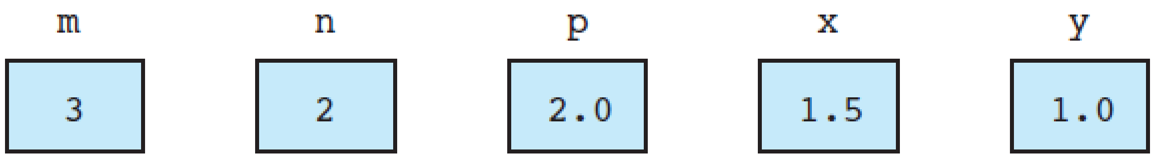
Automatic Type Conversion Rules

$$\begin{tikzcd}
\boxed{\text{char, short}} \arrow[r] & \boxed{\text{int}} \arrow[r] & \boxed{\text{long}} \arrow[r] & \boxed{\text{float}} \arrow[r] & \boxed{\text{double}} \arrow[r] & \boxed{\text{long double}}
\end{tikzcd}$$

* Advice: Avoid automatic type conversion!

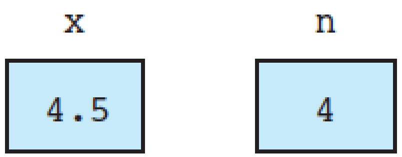
Previously on CEng 230!

```
int  m, n;

double  p , x , y;

 m = 3;
 n = 2;
 p = 2.0;
 x = m / p;     /*  3/2.0  */
 y = m / n;    /*   3/2    */
```



```
x = 9 * 0.5;
n = 9 * 0.5;
```

evaluates to the real number `4.5`. If `x` is of type `double`, the number `4.5` is stored in `x`, as expected. If `n` is of type `int`, only the integral part of the expression value is stored in `n`, as shown.



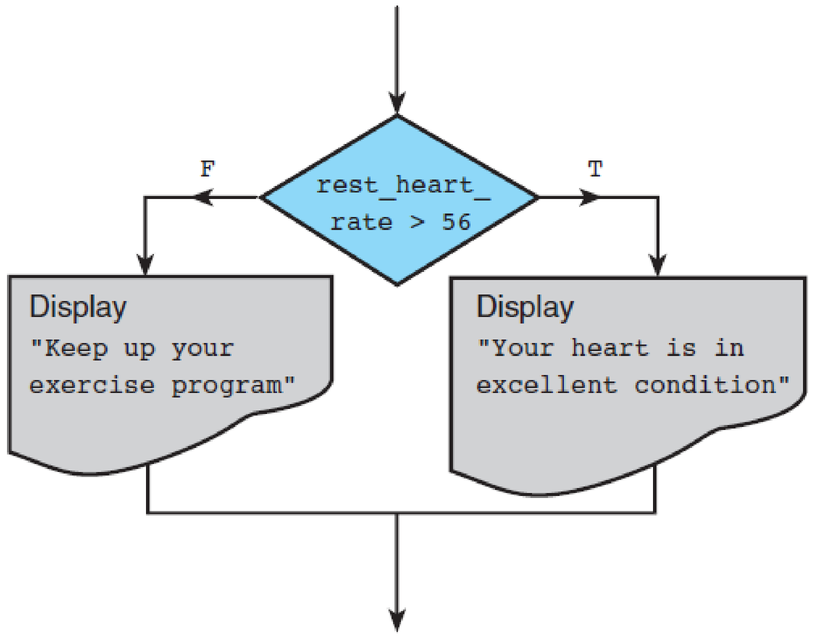
Previously on CEng 230!

Changing the flow of the program

If statements

if statement

```
if (rest_heart_rate > 56)
      printf("Keep up your exercise program!\n");
else
      printf("Your heart is in excellent health!\n");
```



Flowchart of the **if** statement

Changing the flow of the program

Previously on CEng 230!

- if statements

if(expr)
{ ….
}
else if(expr)
{…
}
…
else
{ … }

```
if(a > b)
       printf(“a is bigger”);
else if(a < b)
        printf(“b is bigger”);
else
        printf(“a = b”);
```

Changing the flow of the program

Previously on CEng 230!

- Common mistake with if statements
- **if**(a = 10) { … }
- **if**(a == 10); { … }

```
if (x > 0.0)
pos_prod = pos_prod * x;
```

```
if (crsr_or_frgt == 'C')
  printf("Cruiser\n");
else
  printf("Frigate\n");
```

It displays either Cruiser or Frigate , depending on the character stored in the type char variable crsr_or_frgt .

```
if crsr_or_frgt == 'C'          /* error - missing parentheses */
     printf("Cruiser\n");
printf("Combat ship\n");
```

```
if (crsr_or_frgt == 'C'); /* error - improper placement of ;*/
      printf("Cruiser\n");
printf("Combat ship\n");
```

Previously on CEng230!

Nested if statements

- **if(…)**

  ```
      if( … )
      {....}
  else
  {....}
  ```

nested if statements and alternative decisions

if statement inside another

```
if (x > 0)
      num_pos = num_pos + 1;
else

      if (x < 0)
            num_neg = num_neg + 1;
      else /* x equals 0 */
            num_zero = num_zero + 1;
```

Multiple-Alternative Decision Form of Nested if

SYNTAX:

```
if (condition_1)
    statement_1
else if (condition_2)
    statement_2
        .
        .
        .
else if (condition_n)
    statement_n
else
    statement_e
```

EXAMPLE:

```
/* increment num_pos, num_neg, or num_zero depending
   on x */
if (x > 0)
      num_pos = num_pos + 1;
else if (x < 0)
      num_neg = num_neg + 1;
else /* x equals 0 */
      num_zero = num_zero + 1;
```

Loudness in Decibels (db)	Perception
50 or lower	quiet
51 – 70	intrusive
71 – 90	annoying
91 – 110	very annoying
above 110	uncomfortable

```
/* Display perception of noise loudness */

if (noise_db <= 50)
      printf("%d-decibel noise is quiet.\n", noise_db);
else if (noise_db <= 70)
      printf("%d-decibel noise is intrusive.\n", noise_db);
else if (noise_db <= 90)
      printf("%d-decibel noise is annoying.\n", noise_db);
else if (noise_db <= 110)
      printf("%d-decibel noise is very annoying.\n", noise_db);
else
      printf("%d-decibel noise is uncomfortable.\n", noise_db);
```

Logic error

```
/* incorrect perception of noise loudness */

if (noise_db <= 110)
      printf("%d-decibel noise is very annoying.\n", noise_db);
else if (noise_db <= 90)

      printf("%d-decibel noise is annoying.\n",
              noise_db);
else if (noise_db <= 70)
      printf("%d-decibel noise is intrusive.\n",
              noise_db);
else if (noise_db <= 50)
      printf("%d-decibel noise is quiet.\n",
              noise_db);
else
      printf("%d-decibel noise is uncomfortable.\n", noise_db);
```

Switching values of two variables

```
if (x > y) {                    /* Switch x and y */
      temp = x;                 /* Store old x in temp */
      x = y;                    /* Store old y in x */
      y = temp;                 /* Store old x in y */
}
```

Statement Part	x	y	temp	Effect
	12.5	5.0	?	
`if (x > y) {`				`12.5 > 5.0` is true.
`temp = x;`			12.5	Store old `x` in `temp`.
`x = y;`	5.0			Store old `y` in `x`.
`y = temp;`		12.5		Store old `x` in `y`.

17- What is the output of the following program segment?

```
int x = -1;
if (x++==0)  printf("%d\n",x);
else if(++x>1) printf("%d",x);
else printf("%d",x);
```

a) -1 b) 0 c) 1 d) 2 e) 3

---

18. For what exact range of values of variables a and b, does the following code segment display the value 0?

```
m= -1;
if (a>20)
    if (b<10)
       if (a>=30)
           m =4;
       else
           m=0;
    else
        m=1;
else
    m=2;
printf("%d",m);
```

a) $a > 20$, $b \geq 10$

b) $20 \leq a \leq 30$, $b \leq 10$

c) $20 < a < 30$, $b < 10$

d) $a \geq 30$, $b < 10$

e) $20 < a < 30$, $b \geq 10$

19- Assuming that x,y and flag are integers, what is the value printed by the following if statements?

```
if(x>y)
 if(x>z) printf("%d", x);
 else
  if(z>y) printf("%d",z);
  else printf("%d", y);
else
 if(y>z) printf("%d",y);
 else printf("%d",z);
```

a) minimum b) maximum c) median
d) last e) indeterminate

20- What is the output of the following program segment?

```
int x=6, y=3, A=3, B=5, C=7;
if (x <A && y >B)
   if (y >0)
      printf("A");
   else printf("B");
   else if (y>C || x >0)
         printf("C");
```

a) A b) B c) C d) AC e) no output

27) What will be the output of the program?

Scope of else without {}

```
#include<stdio.h>
void main( ){
   int a = 4;
   if(a == 4)
      printf("a1");
   else
      printf("a2");
           printf("a3");
   printf("a4"); }
```

a) a4   **b)** a1a4   **c)** a2a3a4   **d)** a1a3a4   **e)** a2a3

---

28) What will be the output of the program?

Scope of if without {}

```
#include<stdio.h>
void main()
{
   int a = 9, b = 3;
   if( !a <= 4 )
      b = 5;
      a = 1;
   printf("a=%d b=%d\n", a, b); }
```

a) a = 9, b = 3

b) a = 4, b = 3

c) a = 1, b = 5

d) a = 9, b = 5

e) a = 1, b = 3

32) What will be the output of the program?

```
#include<stdio.h>
void main( ){
   int m=8;
   float n=8.6;
   if (m > n)
      { }
  else {
      m = n * 2;
      n = n / 2; }
     printf(" %d %f ", m, n);
  }
```

a) 17 4.300000
b) 17 4.000000
c) 16 4.300000
d) 16 4.000000
e) Compile error

---

31) What will be the output of the program?

```
#include<stdio.h>
void main( ){
    int z=9;
    z=z-4;
    if( z<9 || ++z>4 ) z=z+2;
    printf(" %d ", z);
    }
```

a) 5 **b)** 6 **c)** 7 **d)** 8 **e)** 9

Today

- More on changing the flow of the program
  - Switch statements
- Defining macros in C
- Sample questions

Conditional Expression Operator

- Conditional expression:
  - Expr ? True-expr : False-expr
  - int a = x > 10 ? 1 : 0;
- Right-to-left associative.
  - X = c ? a : d ? e : f;
- Precedence:
  - c ? X = a : X = b
  - ‘?’ and ‘:’ bracket the expression. True-expr can have operators of any precedence without parentheses.
  - The False-expr part has lower precedence than all operators except ‘=‘ and ‘,’.

y= x > 3 ? a+1 : a-1; means

```
İf (x > 3)
        y=a+1;
else
        y=a-1;
```

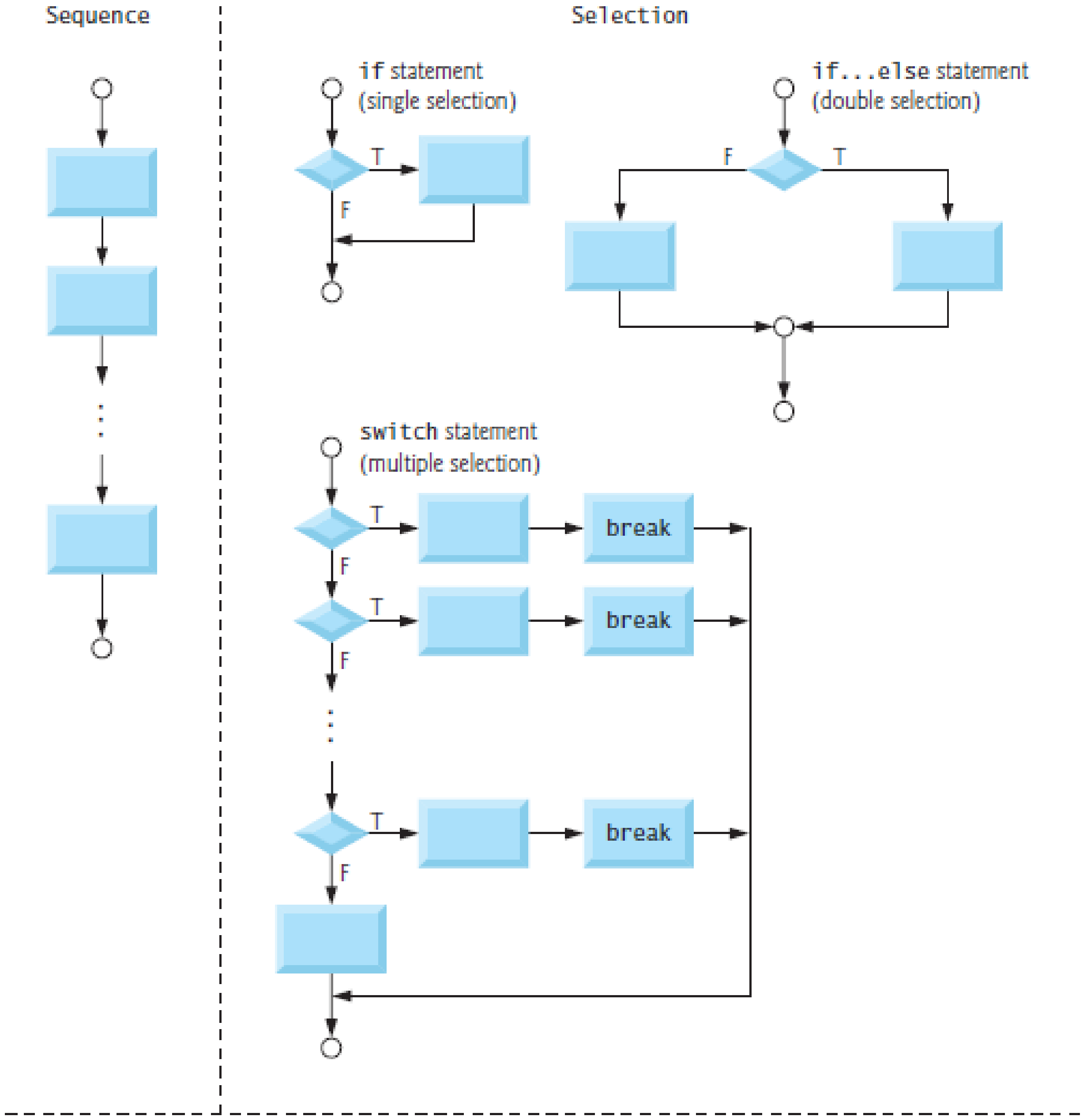
```
z=(a > b) ? a: b;   (finds maximum)
```

```
Printf("%d%c", k, (k%10==9) ? 'A' : 'a');
```

Changing the flow of the program

- Multi-way conditionals: switch statements

```
switch(expr)
{
   case  value-1:
         ….
         break;
   case  value-2:
         ….
         break;
   default:
         ….
         break;
}
```


Sequence
Selection
if statement
(single selection)
T
F
if...else statement
(double selection)
F
T
switch statement
(multiple selection)
T
F
break
T
F
break
T
F
break

Example

-
  ```
  main()
  {
  int i=3;
  switch(i)
  {
  default: printf("zero");
  case 1: printf("one");
  break;
  case 2: printf("two");
  break;
  case 3: printf("three");
  break;
  }
  }
  ```

-
  ```
  main()
  {
  int i=1;
  switch(i)
  {
  default: printf("zero");
  case 1: printf("one");
  case 2: printf("two");
  break;
  case 3: printf("three");
  break;
  }
  }
  ```

Coming back to the elements of C

Simple Macros

- For long and/or frequent constants:
  - #**define** PI 3.14159265

- For long and/or frequent calculations:
  - #**define** Area(Radius) (4*PI*Radius*Radius)
  - … a = 10.0 + Area(2.0);

Remember Types of Errors?

Compile time → Syntax errors

- When the code violated the grammer rules of C.
- Compiler detecs these errors

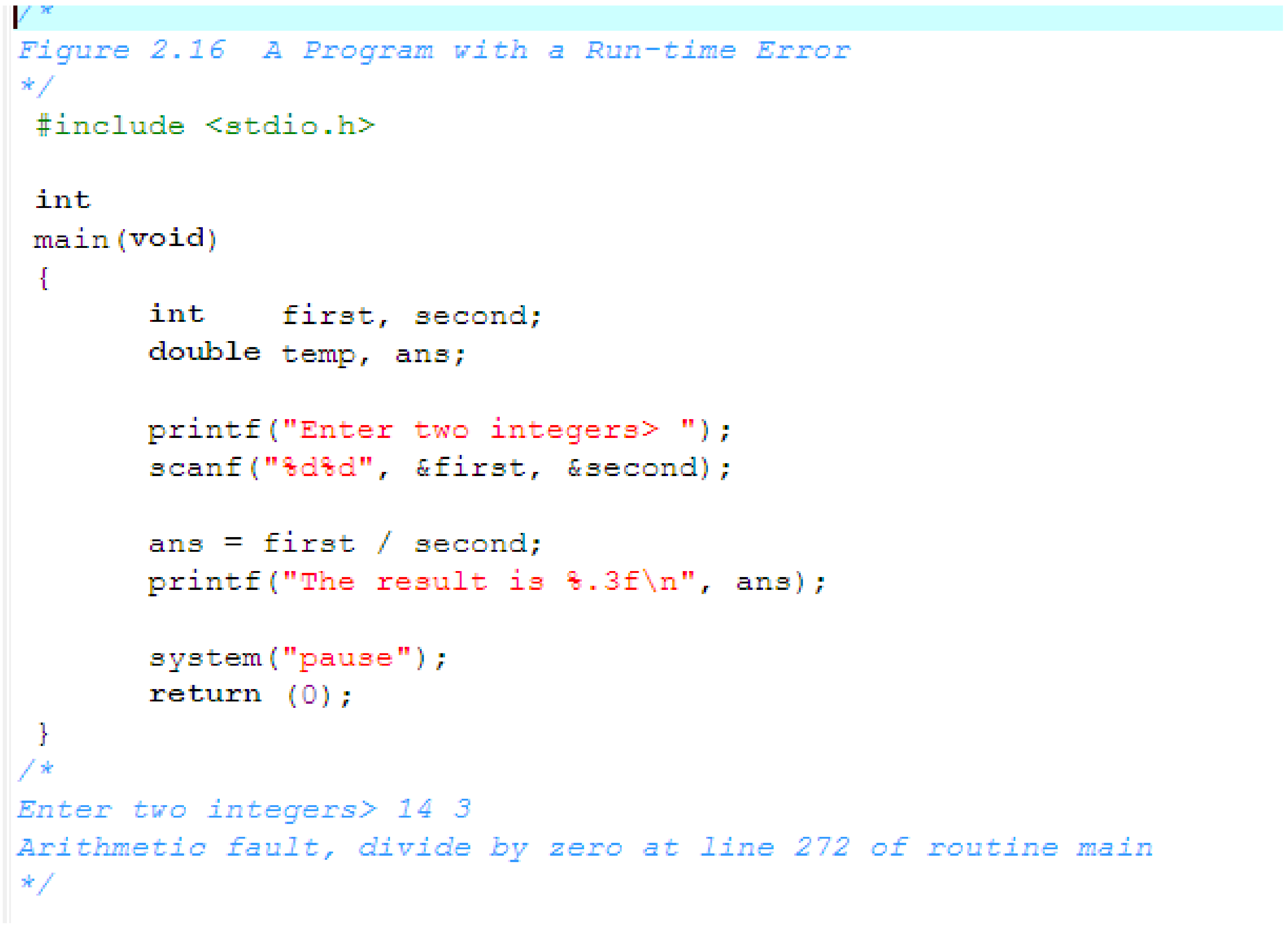
Run-time errors

- happen when the program directs the computer to an illegal operation.
- Such as Division by zero

Logic errors

- A faulty algorithm
- It gives no error message.

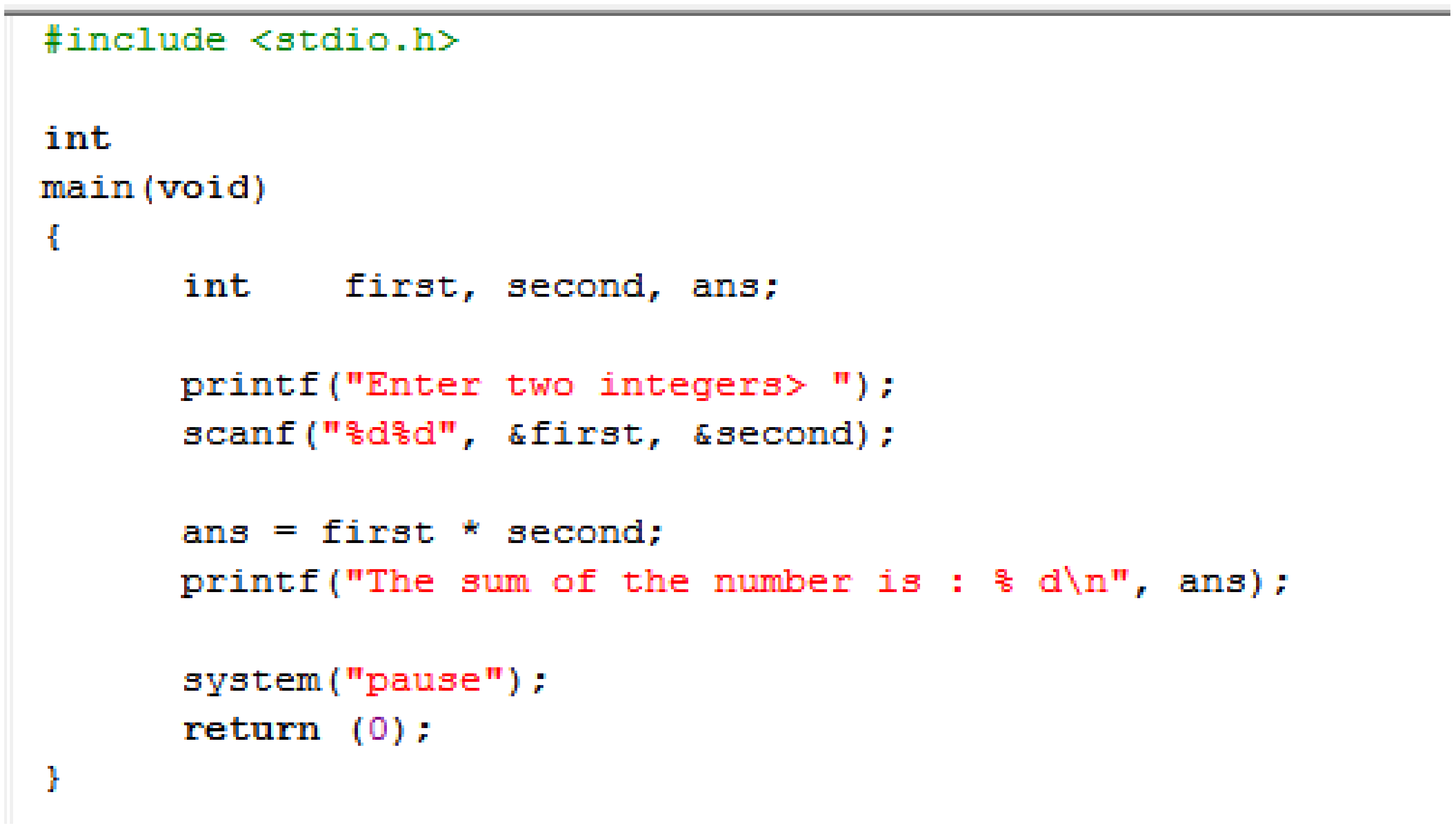
Run-time error (division by zero)



```
/*
Figure 2.16  A Program with a Run-time Error
*/
 #include <stdio.h>

 int
 main(void)
 {
       int    first, second;
       double temp, ans;

       printf("Enter two integers> ");
       scanf("%d%d", &first, &second);

       ans = first / second;
       printf("The result is %.3f\n", ans);

       system("pause");
       return (0);
 }
/*
Enter two integers> 14 3
Arithmetic fault, divide by zero at line 272 of routine main
*/
```

If the value of variable "second" is given as zero

Logic error



```
#include <stdio.h>

int
main(void)
{
        int     first, second, ans;

        printf("Enter two integers> ");
        scanf("%d%d", &first, &second);

        ans = first * second;
        printf("The sum of the number is : % d\n", ans);

        system("pause");
        return (0);
}
```

Sample questions

Evaluate the following expressions with `7` and `22` as operands. (be careful that the values are integers)

```
22 / 7     7 / 22     22 % 7     7 % 22
```

Evaluate the following, assuming that letters have consecutive character codes.

a. `(int)'D' - (int)'A'`
b. `(char)((int)'C' + 2)`
c. `(int)'6' - (int)'7'`

4) What will be the output of the following C program?

```
#include <stdio.h>
#define X 5+3
int main() {
int a = X / 2;
printf("%d", a);
return 0; }
```

a) 2 **b)** 4 **c)** 6 **d)** 8 **e)** 6.5

5) What will be the output of the following C program?

```
#include <stdio.h>
int main() {
  double x, y;
  x = 7;
  x = x / 2;
  y = x + x / 2;
  printf("%.2f %.2f", x, y);
  return 0; }
```

a) 3.00 3.00 **b)** 3.0 4.50 **c)** 3.50 3.50
d) 3.50 5.25 **e)** 4.50 5.25

6) What will be the output of the following C program?

```
  #include <stdio.h>
int
             main
      (void) {
    int a; double b; printf("%d %.2f", a=5+3/2,
b=5+3/2);
    return 0; }
```

a) 6 6.00 **b)** 6 6.50
c) 7 7.00 **d)** 7 7.50
e) This program will not compile successfully because of bad indentation.

```
# include <stdio.h>
int main (void){
printf ("%c,%d,%c,%d",'a','a', 97, 97);
return 0;}
```

a) a,97,a,98 **b)** 97,a,97,a **c)** 97,97,a,a
[✓ **d)** a,97,a,97] **e)** a,97,97,a

9) What will be the output of the following code segment?

```
        double pi= 22/7;
        printf("%3.2f", pi);
```

a)3.142857 **b)**3.142 **c)**3.14 **d)**03.14 [✓ **e)**3.00]

10) What would be the output after execution of the following code?

```
        int x=5,y=3;
        y+=5-y+x++;
        x=y%x;
        printf("%d", x);
```

a) 3 **b)** 5 **c)** 6 [✓ **d)** 4] **e)** 2

12) What would be the output after execution of the following code?

```
int x=2;
double y=22/5*(double)x;
printf("%.2f", y);
```

a) 2.20 **b)** 2.00 [✓ **c)** 8.00] **d)** 8.80 **e)** 2.80

16) What would be the output after execution of the following code?

10 is wrong!

```
int b,a=3,c=5;
b=12+a--/++c-(--a);
printf("%d",b);
```

[✓ **a)** 10] **b)** 12 **c)** 9 **d)** 11 **e)** 8

6) What could be the output of the following code segment?

printf("%07.4f", 22/7.0);

a) 03.143
b) 003.143
c) .314285
[✓ **d)** 03.1429]
e) .003143

15) What will be the output of the following code segment?

```
int a=4,b=3;
a= 4*3-2+b--/2*3%2*4-2;
printf("%d",a--);
```

a) 8 **b)** 7 [✓ **c)** 12] **d)** 10 **e)** 11

Hint: b--/2*3%2*4 : execute this part from left to right

2) What will be the output after the input of 13 ?

```
int s;
scanf("%d", &s);
printf ("%d", s%2+--s);
```

This one is first executed

a) 12 **b)** 6 **c)** 7 **d)** 8 [✓ **e)** None of them]

4) What will be the output after the input of 7?

```
int x;
printf ("Enter a number");
scanf("%d",&x);
printf("%d %d %d", x - 1, x, x--);
```

a) 5 6 7 **b)** 4 6 6 **c)** 6 7 6 [✓ **d)** 6 7 7] **e)** 4 5 6

4) If *a* is 5, *b* is 4 , *c* is 10 what is the output?

```
a=b=c+6%2;
printf("%d %d %d", a,b,c);
```

a) `5 10 10` **b)** `13 13 10` **c)** `10 10 10` **d)** `5 4 10` **e)** `13 10 10`

5) What is the output of the below code segment ?

```
int i=32;
char c;
c=i;
printf("%d", c);
```

a) 23 **b)** 'c' **c)** 69 **d)** 'E' **e)** 32

8) What is the C equivalent of the following expression?

$$x = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

a) x=(-b-sqrt(b*b-4*a*c))/(2*a)
b) x=(-b-sqrt(b*2-4*a*c)/2*a
c) x=(-b-sqrt(b*2-4ac)/2a
d) x=((-b)-sqrt(b*2-4ac)/2a
e) x=-b-sqrt(b*b-4*a*c)/2*a