ESc101: Decision making using if-else and switch statements

Instructor: Krithika Venkataramani

Semester 2, 2011-2012

The content of many of these slides are taken from the Lecture slides of Prof. Arnab Bhattacharya and Prof. R. K. Ghosh

Writing Simple C Programs

- Use standard files having predefined instructions
  - stdio.h: has defined standard input and output instructions
    - always needed for reading input /displaying output
  - math.h: has specific math instructions such as square-root, power
    - not needed if these instructions are not used

```
#include<stdio.h>
#include<math.h>
```

- main function has the program

```
void main()
{
    ---
}
```

Writing Simple C Programs

- Declare variables to use/process different data types

```
int number;
float real;
char letter;
```

  - Can assign a constant as initial value of the variables

```
int number = 5;
float real = 5.5;
char letter = 'A';
```

- Use printf for displaying output on monitor
  - printf("\nInteger = %d", number);
    Integer = 5
  - %d is a place holder (format specifier) for displaying the value of the integer variable number
  - \n :moves to a new line while displaying

Writing Simple C Programs

- Use the appropriate format specifier for displaying different variable types using printf
  - printf("\nReal number = %f", real);
    Real number = 5.500000
  - printf("\nCharacter = %c and Integer = %d", letter, number);
    Character = 'A' and Integer = 5

Type	Format Specifier
char	%c
int	%d
unsigned int	%u
float	%f, %g, %e
double	%lf
long double	%Lf

Writing Simple C Programs

- Use scanf for reading input from keyboard
- scanf requires & before the variable name
  - Why it is required will be explained later
- Examples
  - scanf("%d", &number);
  - scanf("%f", &real);
  - scanf("%c", &letter);
  - Use appropriate format specifiers for different variable types

Type	Format Specifier
char	%c
int	%d
unsigned int	%u
float	%f, %g, %e
double	%lf
long double	%Lf

if-else statements

- Used in Decision making
- Example Algorithm: Find the minimum of two integers
  1. Compare the two integers x and y
  2. If x < y, then min = x
  3. Otherwise, min = y
- To capture the above logic in C, if-else statements are used

```
if ( condition )
{
    statements1
}
else
{
    statements2
}
```

- Entire if-else is a single statement

Program to find sum and minimum of two numbers

```
# include <stdio.h>
void main ()
{
    int x, y;
    int min, sum;
    scanf (``%d", &x);
    scanf (``%d", &y);
    sum = x + y;
    if (x < y)
    {
        min = x;
    }
    else
    {
        min = y;
    }
    printf (``Minimum is %d and Sum is %d\n", min, sum);
}
```

Understanding if-else statement

```
if ( condition )
{
        statements1
}
else
{
        statements2
}
```

- Condition must evaluate to a boolean value
- When condition is ‘true’, if-statement is executed
- When condition is ‘false’, else-statement is executed
- Any expression fits as a condition
- else- part can be omitted

```
if ( condition )
{
        statements1
}
```

Understanding if-else statement

- A block of statements may be used in if and else part
  - A block of statements is equivalent to a single statement

```
if ( condition )
{
statement1
statement2
}
else
{
statement3
statement4
}
```

Nested if-else

- Else with more than one previous if is ambiguous

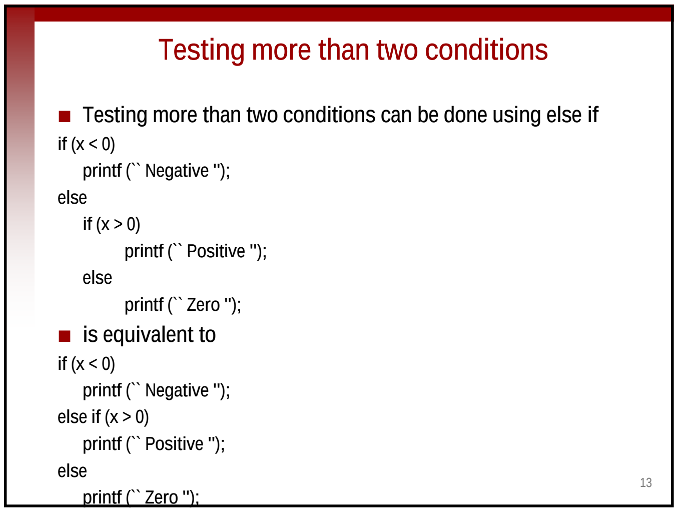
```
if ((x + y) > 0)
    if (x < y)
        printf (``x is minimum ");
    else
        printf (``y is minimum ");
```

- Rule: else is associated with nearest if
- Indenting lines in program helps in understanding

Nested if-else

- Use braces if intended otherwise

```
if ((x + y) > 0)
{
if (x < y)
    printf (``x is minimum ");
}
else
    printf (``x + y is negative ");
```



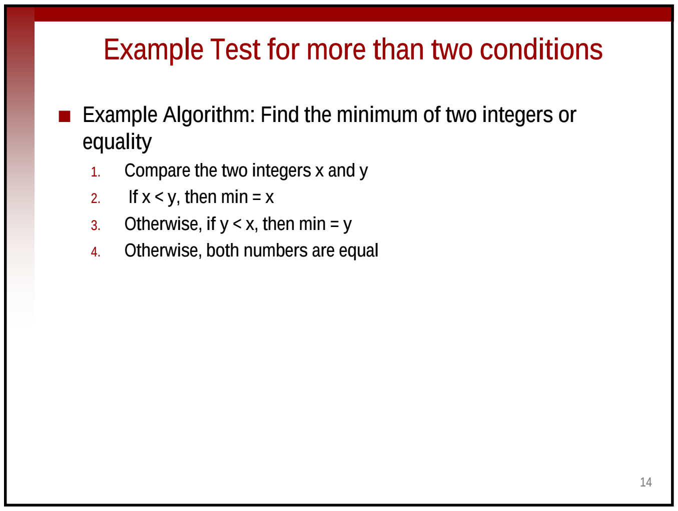
Testing more than two conditions

- Testing more than two conditions can be done using else if

```
if (x < 0)
    printf (`` Negative ");
else
    if (x > 0)
          printf (`` Positive ");
    else
          printf (`` Zero ");
```

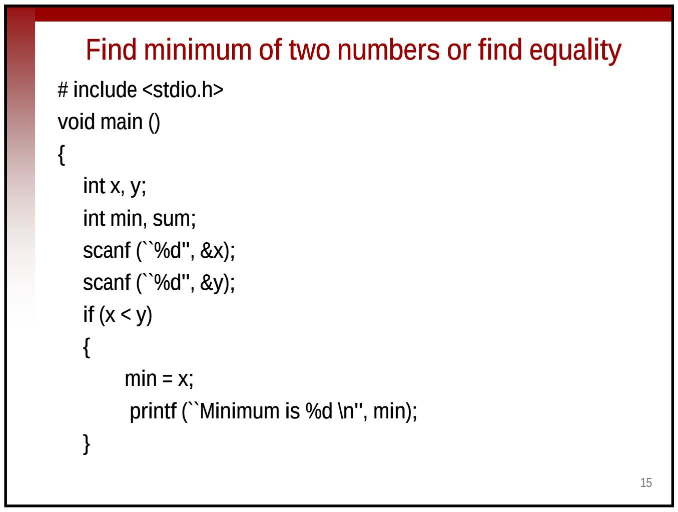
- is equivalent to

```
if (x < 0)
    printf (`` Negative ");
else if (x > 0)
    printf (`` Positive ");
else
    printf (`` Zero ");
```



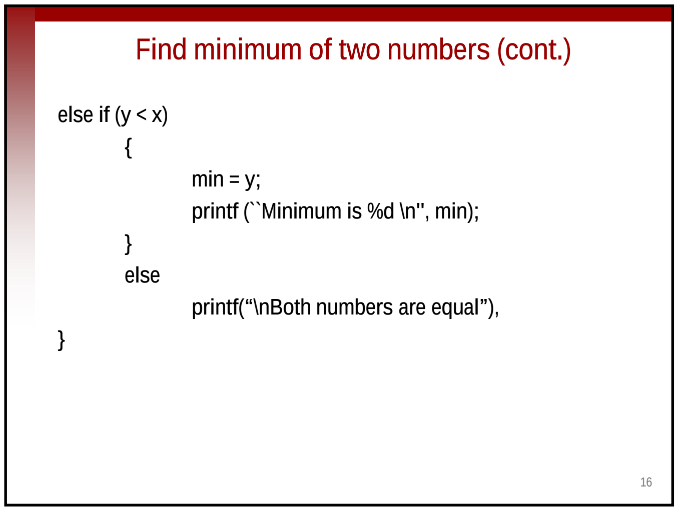
Example Test for more than two conditions

- Example Algorithm: Find the minimum of two integers or equality
    1. Compare the two integers x and y
    2. If x < y, then min = x
    3. Otherwise, if y < x, then min = y
    4. Otherwise, both numbers are equal



Find minimum of two numbers or find equality

```
# include <stdio.h>
void main ()
{
    int x, y;
    int min, sum;
    scanf (``%d", &x);
    scanf (``%d", &y);
    if (x < y)
    {
          min = x;
           printf (``Minimum is %d \n", min);
    }
```



Find minimum of two numbers (cont.)

```
else if (y < x)
          {
                    min = y;
                    printf (``Minimum is %d \n", min);
          }
          else
                    printf("\nBoth numbers are equal"),
}
```

Sample program to find triangle type

- Please take the 3 sides of a triangle, and print whether the triangle is an equilateral, isosceles or scalene triangle.

```
#include<stdio.h>
void main()
{
float  side1, side2, side3; //declare variables to take the 3 sides
    of a triangle
printf("Enter the three sides of a triangle: ");
scanf("%f  %f  %f", &side1, &side2, &side3);
if ( ((side1+side2)>side3) && ((side2+side3)>side1) &&
    ((side1+side3)>side2) )
     {
```

Program to find type of triangle (cont.)

```
     if ( (side1==side2) && (side1==side3) )
        printf(" \nThe triangle is equilateral");
     else if ((side1!=side2) && (side2!=side3) && (side1!=side3))
        printf("\nThe triangle is scalene");
     else
        printf("\nThe triangle is isosceles");
     }
else
    printf("\nA triangle is not formed using these sides");
}
```

Lab 1 : Q1 sample solutions

- Take a character as input from the user. Check whether the character is an alphabet or not.
- Algorithm:

1. Input a character
2. If character is between 'a' to 'z', or between 'A' to 'Z', it is an alphabet
3. Otherwise, it is not an alphabet

Lab 1 sample solutions: Q1

```
#include<stdio.h> /* Q1. Author:rahule@cse.iitk.ac.in  */
int main()
{
   char ip;
   printf("Enter the character to be checked: ");
   scanf("%c",&ip);
   //checking if it is a Alphabet
   if( (ip>='A'&&ip<='Z') || (ip>='a'&&ip<='z') )
   {
      printf("The input character is an alphabet\n");
   }
   else
   {
      printf("The input character is NOT an alphabet\n");
   }
}
```

Lab 1: Q2 Sample Solutions

- Take as input 4 numbers. Print arithmetic mean & harmonic mean. Print the maximum of the two means.
- Algorithm

1. Input 4 real numbers: a, b, c, d
2. If any of the numbers is not positive, harmonic mean is not valid
3. Otherwise, 1/(harmonic mean) = ((1/a) + (1/b) + (1/c) + (1/d))/4
4. Arithmetic mean = (a+b+c+d)/4
5. If harmonic mean is valid and harmonic mean > arithmetic mean, max = harmonic mean
6. Otherwise, max = Arithmetic mean

Lab 1: Q2 Sample Solutions

```
/* Q2. Author:rahule@cse.iitk.ac.in */
#include<stdio.h>
int main()
{
   float n1,n2,n3,n4,arithmetic_mean=0,harmonic_mean;
   int flag = 0;
   scanf("%f%f%f%f", &n1,&n2,&n3,&n4);
   //calculating the arithmetic mean
   arithmetic_mean=(n1+n2+n3+n4)/4;
   printf("Arithmetic Mean: %f\n",arithmetic_mean);
   //calculating the harmonic mean
   if(n1 <= 0 || n2 <= 0 || n3 <= 0 || n4 <= 0 )
     printf("Harmonic mean can not be calculated as atleast one number is not
     positive!\n");
```

Lab 1: Q2 Sample Solutions (cont.)

```
else
   {
     flag = 1;
      harmonic_mean=4/(1/n1 + 1/n2 + 1/n3 + 1/n4);
      printf("HarmonicMean: %f\n",harmonic_mean);
   }
 //checking which one is maximum
   if((flag ==1)&&(arithmetic_mean= =harmonic_mean))
      {
            printf("Harmonic Mean is equal to arithmetic meanr\n");
            printf("Maximum mean = %f",harmonic_mean);
     }
   else //prints Arithmetic mean is larger even if harmonic mean is not valid
      {
            printf("Arithmetic Mean is larger\n");
            printf("Maximum mean = %f",arithmetic_mean);
     }
}
```

Q2 alternate sample solution using nested if-else

```
   //calculating the harmonic mean
   if(n1 <= 0 || n2 <= 0 || n3 <= 0 || n4 <= 0 )
   {
     printf("Harmonic mean can not be calculated!\n");
   }
   else
   {
      harmonic_mean=4/(1/n1 + 1/n2 + 1/n3 + 1/n4);
      //printing the results
      printf("HarmonicMean: %f\n",harmonic_mean);
      //checking which one is maximum
      if(arithmetic_mean>harmonic_mean)
         printf("Arithmetic Mean is larger\n");
      else
        printf("Harmonic Mean is equal to Arithmetic meanr\n");
   }
}
```

Lab 1: Q3 sample solution using if statement

- Take a 5 digit integer as input from the user. Count the total number of zeroes in it and print the result.
- Algorithm:

1. Input the integer
2. Initialize zero_count to 0
3. Find the remainder of integer by dividing using 10
4. If remainder is zero, then increment zero_count by 1
5. Divide the integer by 10
6. Use the quotient as the new integer
7. Repeat Steps 3 to 5 an additional 4 times
8. Display zero_count

25

Lab 1: Q3 sample solution using if statement

```
/*author:rahule@cse.iitk.ac.in*/
#include<stdio.h>
int main()
{
   int n, count=0;
   printf("Enter the FIVE DIGIT integer\n");
   scanf("%d",&n);
   if  ( ((n<=99999)&&(n>=10000))  ||    ((n>=-99999)&&(n<=-10000))  )
    { //check for zeroes only if it is a 5 digit integer
           if(n%10 == 0) //checking if the 5th(last) digit is zero
                     count++;
           n=n/10;   //converting to a 4 digit integer
           if(n%10 == 0) // checking if 4th digit  of original integer is zero
                     count++;
           n=n/10; //converting to a 3 digit integer
```

26

Lab 1: Q3 sample solution using if statement (cont.)

```
     //checking if the 3rd digit is zero
    if(n%10 == 0)
              count++;
    n=n/10;
    //checking if the 2nd digit is zero
    if(n%10 == 0)
              count++;
    n=n/10;
    }// end of if condition for checking a 5 digit integer
//printing the results
printf("Number of zeros: %d\n",count);
}
```

27

Multiple if-else

- Consider

```
if ( section == 1)
     printf (`` TB101 ");
else if ( section == 2)
     printf (`` TB102 ");
else if ( section == 12)
     printf (`` TB112 ");
else
     printf (`` Wrong section ");
```

- Multiple else-if statements are better written using switch statements
- ‘switch’ works only when the same variable is tested for equality against different constant values

28

Switch used for multi-way decision

```
switch (expression)
{
        case constant-expression1: statements; break;
        case constant-expression2: statements; break;
        default: statements; break;
}
```

- switch is useful when multiple decisions can be made depending on the value of the expression
- The expression must evaluate to a constant integer
  - The case values are constant integers
  - Characters are mapped to integers and can be used in switch
  - Real numbers (float, double) cannot be used in switch
- default is executed when variable evaluates to none of the other values
- break brings the control out of the switch statement

29

Switch statement

- Example

```
switch ( section )
{
case 1: printf (`` TB101 "); break ;
case 2: printf (`` TB102 "); break ;
case 12: printf (`` TB112 "); break ;
default : printf (`` Wrong section "); break ;
}
```

30

Switch statement

- Important: Without break, next case is also executed

```
switch (x)
{
case 0: printf (``0");
case 1: printf (``1");
default : printf (``2");
}
```

- When x is 0, all of 0, 1 and 2 are printed
- When x is 1, both 1 and 2 are printed

31

Switch statement without break

- switch case without break is useful when same statement needs to be executed for multiple cases
- Suppose there are two sections, 1 and 2, on Monday, two sections, 3 and 4, on Tuesday, and others on Wednesday
- Output the day based on input section

```
switch ( section )
{
case 1: ;
case 2: printf (`` Monday "); break ;
case 3: ;
case 4: printf (`` Tuesday "); break ;
default : printf (`` Wednesday "); break ;
}
```

32

break brings control out of switch statement

- Control is transferred to the case statement depending on the value of the expression
- Control is transferred to default case when the value of the expression does not match any of the case values
- Without break, the statements in the next case are also executed
- While break is not required for the last case (could be the default case), it is a good programming practice as its useful when additional cases are inserted
- Removing break is sometimes useful when the same statement needs to be executed for multiple cases

33

switch used for multiple options in menu selection

```
------
printf("Travel guide\n");
printf("A: Air/flight timings\n");
printf("T: Train timings\n");
printf("B: Bus timings\n");
printf("Enter your choice: ");
scanf("%c", &character);
switch  (character)
{
      case 'A': air_display(); break; //Using a function to display flight times
      case 'T': train_display(); break; //using a function to display train times
      case 'B': bus_display(); break; //using a function to display bus times
      default : printf("No choice made");
}
-----
```

34

break not used when multiple cases need same statement

```
/*display color name based on first character of color (small or
    capital letters)*/
scanf("%c", &color)
switch (color)
{
      case 'w': case 'W':  //for both 'w' and 'W', "White" is  displayed
      printf("White\n"); break;
      case 'r':  case 'R': //for both 'r' and 'R', "Red" is  displayed
      printf("Red\n"); break;
      case 'g': case 'G': //for both 'g' and 'G', "Green" is  displayed
      printf("Green\n"); break;
      default : printf("Choose among known colors\n");
}
```

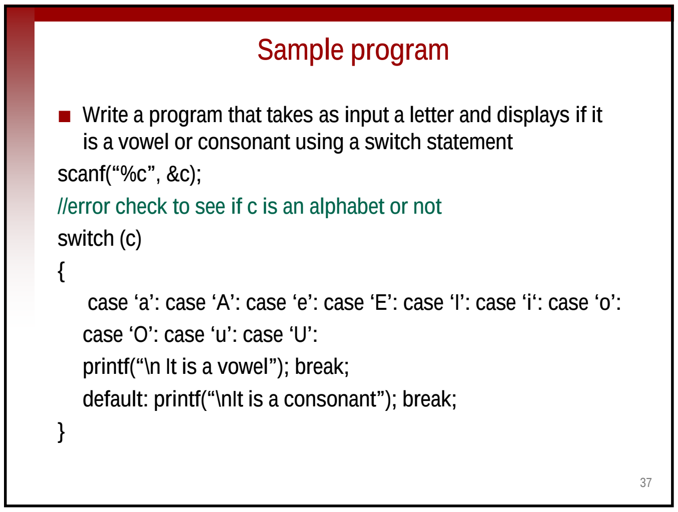
35

More on Switch statement

- The case values in switch are to be constant integers
- break brings execution out of the switch statement
- For the same statement s to be executed for multiple cases, put the statements in the last of these cases and leave the rest of the case values blank with no break

```
switch (color)
{
      case 'w': case 'W':  //for both 'w' and 'W', "White" is  displayed
      printf("White\n"); break;
}
switch (color)
{
      case 'w' || 'W':  printf("White\n"); break; //does not provide desired
}      //result as  'w' || 'W' = 1 and this case is equivalent to case 1
```

36

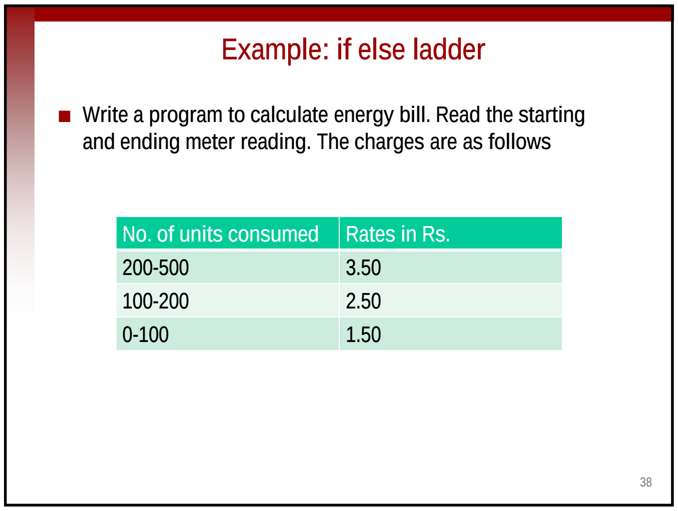


Sample program

- Write a program that takes as input a letter and displays if it is a vowel or consonant using a switch statement

```
scanf("%c", &c);
//error check to see if c is an alphabet or not
switch (c)
{
     case 'a': case 'A': case 'e': case 'E': case 'I': case 'i': case 'o':
    case 'O': case 'u': case 'U':
    printf("\n It is a vowel"); break;
    default: printf("\nIt is a consonant"); break;
}
```

37

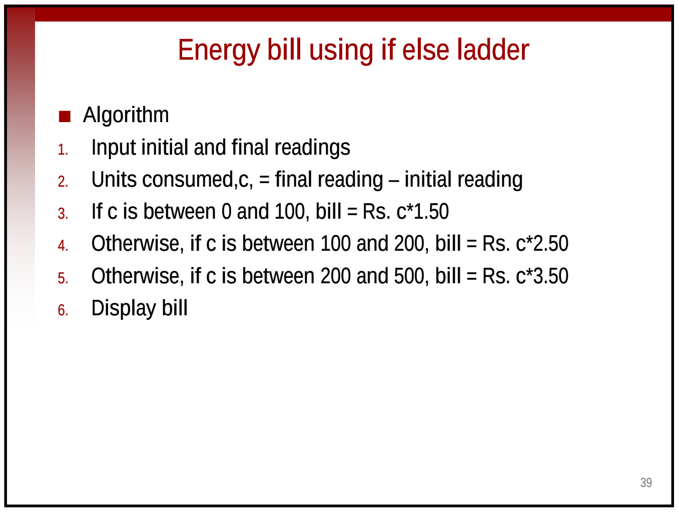


Example: if else ladder

- Write a program to calculate energy bill. Read the starting and ending meter reading. The charges are as follows

No. of units consumed	Rates in Rs.
200-500	3.50
100-200	2.50
0-100	1.50

38

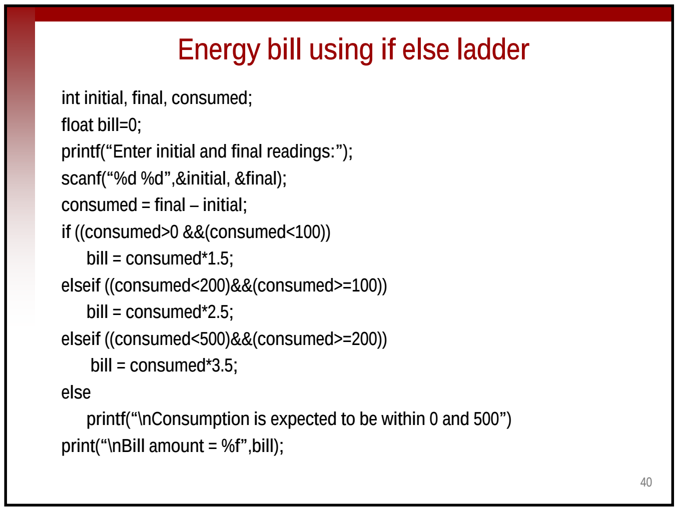


Energy bill using if else ladder

- Algorithm

```
1. Input initial and final readings
2. Units consumed,c, = final reading – initial reading
3. If c is between 0 and 100, bill = Rs. c*1.50
4. Otherwise, if c is between 100 and 200, bill = Rs. c*2.50
5. Otherwise, if c is between 200 and 500, bill = Rs. c*3.50
6. Display bill
```

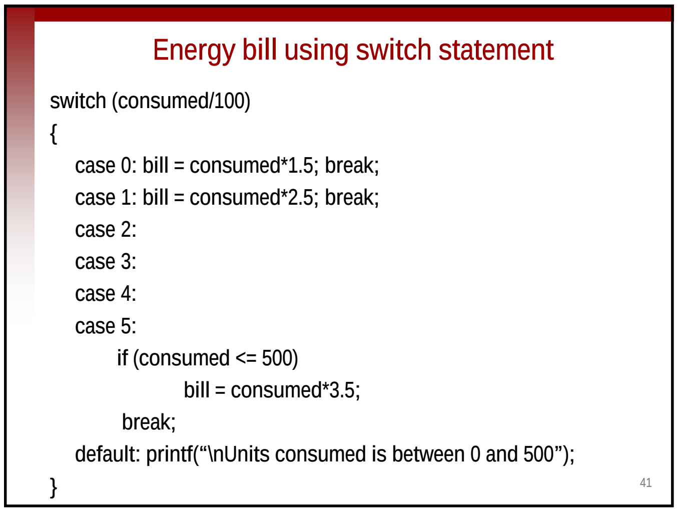
39



Energy bill using if else ladder

```
int initial, final, consumed;
float bill=0;
printf("Enter initial and final readings:");
scanf("%d %d",&initial, &final);
consumed = final – initial;
if ((consumed>0 &&(consumed<100))
    bill = consumed*1.5;
elseif ((consumed<200)&&(consumed>=100))
    bill = consumed*2.5;
elseif ((consumed<500)&&(consumed>=200))
     bill = consumed*3.5;
else
    printf("\nConsumption is expected to be within 0 and 500")
print("\nBill amount = %f",bill);
```

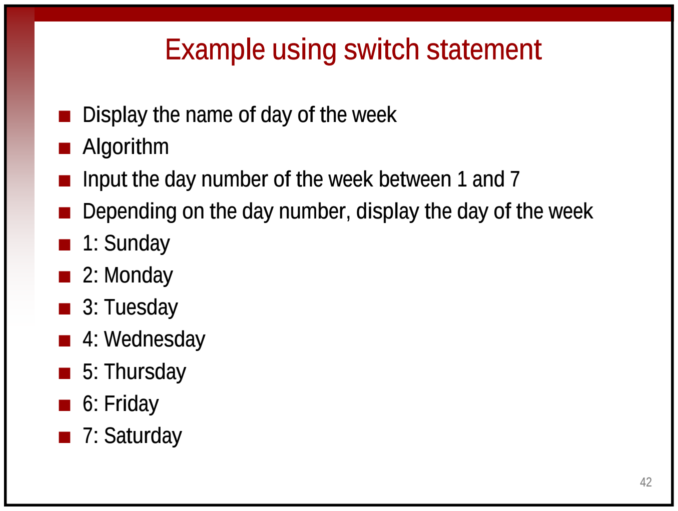
40



Energy bill using switch statement

```
switch (consumed/100)
{
    case 0: bill = consumed*1.5; break;
    case 1: bill = consumed*2.5; break;
    case 2:
    case 3:
    case 4:
    case 5:
           if (consumed <= 500)
                      bill = consumed*3.5;
            break;
    default: printf(“\nUnits consumed is between 0 and 500”);
}
```

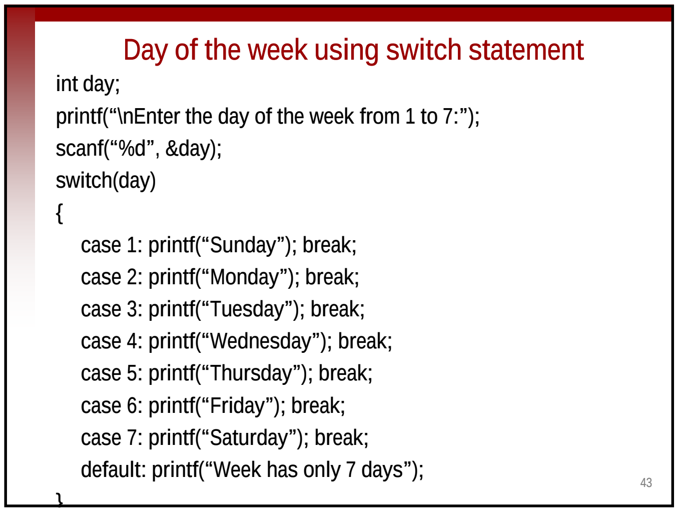
41



Example using switch statement

- Display the name of day of the week
- Algorithm
- Input the day number of the week between 1 and 7
- Depending on the day number, display the day of the week
- 1: Sunday
- 2: Monday
- 3: Tuesday
- 4: Wednesday
- 5: Thursday
- 6: Friday
- 7: Saturday

42



Day of the week using switch statement

```
int day;
printf(“\nEnter the day of the week from 1 to 7:”);
scanf(“%d”, &day);
switch(day)
{
    case 1: printf(“Sunday”); break;
    case 2: printf(“Monday”); break;
    case 3: printf(“Tuesday”); break;
    case 4: printf(“Wednesday”); break;
    case 5: printf(“Thursday”); break;
    case 6: printf(“Friday”); break;
    case 7: printf(“Saturday”); break;
    default: printf(“Week has only 7 days”);
```

43