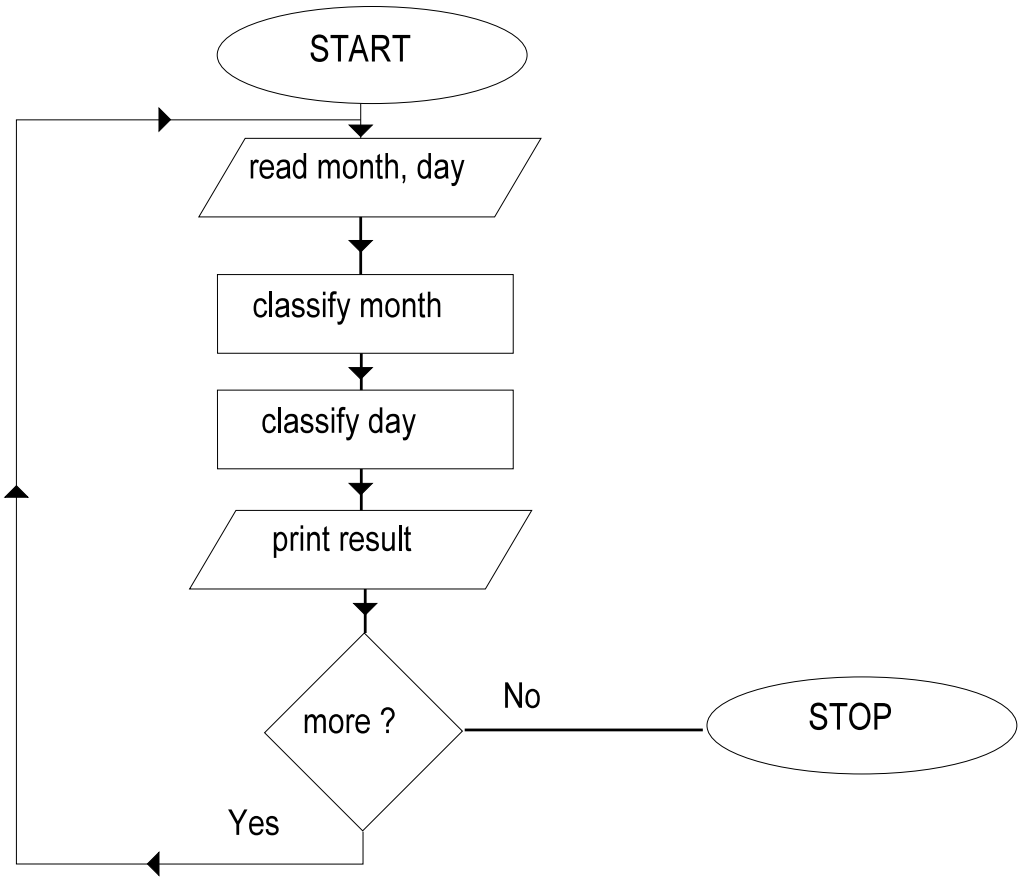
Control Structures

- **Problem:**

  Write a C program that will read in two integer values which will be interpreted as a month (1 to 12) and a day within the month respectively. The program should classify these two pieces of data in terms of which season the month is in and which week of the month the day is in. The program should continue to read month-day combinations until the entire set of data has been processed.

  For simplicity assume:

  Winter: December, January, February
  Spring: March, April, May
  Summer: June, July, August
  Fall: September, October, November



Pseudocode

- **Pseudocode for the Problem:**
  read a month and a day
  classify the month and day
  print the results
  repeat the above where there is more data to process

- **Typical Output:**
  month = 11 day = 14
  month 11 is November
  Fall is the season
  2 is the week number for day 14

- **Revised Pseudocode for the Problem:**
  read a month and a day
  *call a function **classify** which will classify the month and day*
  and print the results.
  repeat the above where there is more data to process

- **Pseudocode for the Function 'classify':**
  *call a function **translate** to translate the month to a string*
  *call a function **whichseason** to classify the month*
  *call a function **whichweek** to classify the day*
  print the results of the function calls
  return control to the calling program

- **Pseudocode for the Function 'whichseason':**
  classify the month into one of four seasons
  print the results
  return control to the calling program

- **Pseudocode for the Function 'whichweek':**
  classify the day into one of five weeks
  (assume 1...7 = week 1; 8...14 = week 2, etc.)
  print the results
  return control to the calling program

- **First Draft of the Program:**

```
/*  program to classify month-day combinations
 *  in terms of season and weeks
 */
#include < stdio.h>

// function prototypes go here

void main()
{
    int month,day;

    // prompt for the month and day
    printf("\nType in the number of a month "
        "(1-January ... 12-December): ");
    scanf("%d",&month);
    printf("\nType in the a date within the month (1-31): ");
    scanf("%d",&day);
    printf("\nmonth =  %d day =  %d\n",month,day);

    // call a function to classify the month-day combination
    classify (month,day);

    ...
}

// functions go here
```

Detecting the End of a Set of Data

- **User-Response Method:**

```
printf("\n\nType Y to continue; N to stop: ");
scanf(" %c",&answer);       //Note 'SPACE' before the %c
```

- **Revised Main Program:**

```
/*  program to classify month-day combinations
 *  in terms of season and weeks
 */
#include < stdio.h>

// function prototypes go here

void main()
{
   int month,day;
   char answer;

   // prompt for the month and day
   printf("\nType in the number of a month "
      "(1-January ... 12-December): ");
   scanf("%d",&month);
   printf("\nType in the a date within the month (1-31): ");
   scanf("%d",&day);
   printf("\nmonth =  %d day =  %d\n",month,day);

   // call a function to classify the month-day combination
   classify (month,day);

   // prompt whether or not to continue
   printf("\n\nType Y to continue; N to stop: ");
   scanf(" %c",&answer);
   if answer is Y repeat the above process
   if answer is N stop
}
```

do-while Loops

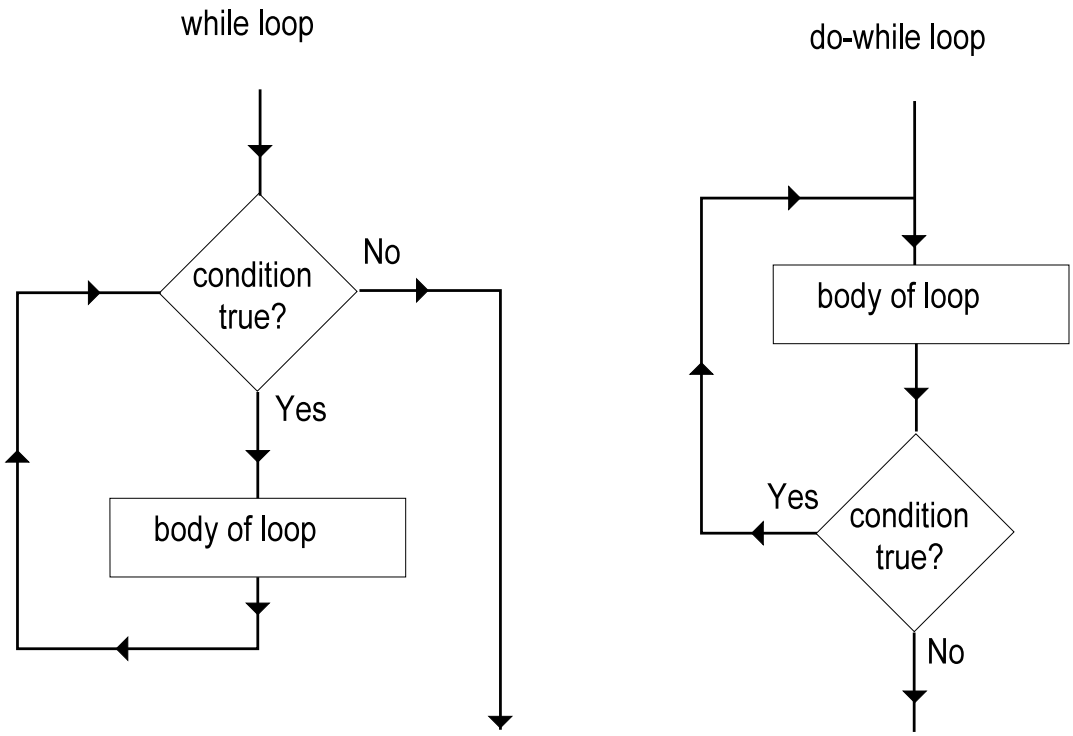
- **Review of while loop:**

```
while (condition)
    body of the loop
```

- **do-while statement (do-while loop):**

```
do
    body of the loop
while (condition);
```

- **Comparison of while and do-while loops:**



- **Next Revision of Main Program:**

```
/*  program to classify month-day combinations
 *  in terms of season and weeks
 */
#include < stdio.h>

// function prototypes go here

void main()
{
   int month,day;
   char answer;

   do {
      // prompt for the month and day
      printf("\nType in the number of a month "
         "(1-January ... 12-December): ");
      scanf("%d",&month);
      printf("\nType in the a date within the month (1-31): ");
      scanf("%d",&day);
      printf("\nmonth =  %d day =  %d",month,day);

      // call a function to classify the month-day combination
      classify (month,day);

      // prompt whether or not to continue
      printf("\n\nType Y to continue; N to stop: ");
      scanf(" %c",&answer);
   } while (answer = =  'Y');
}
```

- **The Function classify:**

```
/*  Function classify
 *  Input:
 *      month - a number specifying a month in the year
 *      day - a number specifying a day within a month
 *  Process:
 *      calls a function to translate the month into a string
 *      calls a function to determine which season month is in
 *      calls a function to determine which week of the month
 *      day is in.
 *      Illegal entries for month and day will be caught
 *  Output:
 *      prints the results of the classification
 */
void classify(int month, int day)
{
    // test for valid month
    if (month <  1 OR month >  12)
        // invalid month entered
        printf("\n%d is not a valid value for the month",month);
    else {
        // valid month entered
        translate(month);               //translate integer to string
        whichseason(month);
    }

    // test for valid day
    if (day <  1 OR day >  31)
        // invalid day entered
        printf("\n%d is not a valid value for the day",day);
    else
        // valid day entered
        whichweek(day);

    return;
}
```

Logical Operators

- **Logical Operators (&&, ||, !):**
  && means logical AND
  || means logical OR
  ! means logical NOT

- **Truth Tables for the Logical Operators:**

a	b	a **&&** b	a **||** b	**!**a	**!**b
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	F
F	F	F	F	T	T

- **Order of Precedence:**
  a) !
  b) &&
  c) ||

- **Handling Illegal Data Values:**

```
// test for valid month
if (month <  1 OR month >  12)
    // invalid month entered
    printf("\n%d is not a valid value for the month",month);
else {
    ...
}

// test for valid month
if (month <  1 || month >  12)
    // invalid month entered
    printf("\n%d is not a valid value for the month",month);
else {
    ...
}
```

- **The Revised Function classify:**

```
/*  Function classify
 *  Input:
 *      month - a number specifying a month in the year
 *      day - a number specifying a day within a month
 *  Process:
 *      calls a function to translate the month into a string
 *      calls a function to determine which season month is in
 *      calls a function to determine which week of the month
 *          day is in.
 *      Illegal entries for month and day will be caught
 *  Output:
 *      prints the results of the classification
 */
void classify(int month, int day)
{
    // test for valid month
    if (month <  1 || month >  12)
        // invalid month entered
        printf("\n%d is not a valid value for the month",month);
    else {
        // valid month entered
        translate(month);               //translate integer to string
        whichseason(month);
    }

    // test for valid day
    if (day <  1 || day >  31)
        // invalid day entered
        printf("\n%d is not a valid value for the day",day);
    else
        // valid day entered
        whichweek(day);

    return;
}
```

- **The Function translate:**

```
/*  Function translate
 *  Input:
 *      month - a number specifying a month in the year
 *  Process:
 *      translates the month into a string
 *  Output:
 *      prints the name associated with month
 */
void translate(int month)
{
    printf("\nmonth %d is ",month);
    if (month ==  1) printf("January");
    if (month ==  2) printf("February");
    if (month ==  3) printf("March");
    if (month ==  4) printf("April");
    if (month ==  5) printf("May");
    if (month ==  6) printf("June");
    if (month ==  7) printf("July");
    if (month ==  8) printf("August");
    if (month ==  9) printf("September");
    if (month ==  10) printf("October");
    if (month ==  11) printf("November");
    if (month ==  12) printf("December");
    return;
}
```

- **The Function whichseason:**

```
/*  Function whichseason
 *  Input:
 *      month - a number specifying a month in the year
 *  Process:
 *      determines which season month is in
 *  Output:
 *      prints the name of the season
 */
void whichseason(int month)
{
    if (month == 12 || month == 1 || month == 2)
        printf("\nWinter is the season");
    if (month == 3 || month == 4 || month == 5)
        printf("\nSpring is the season");
    if (month == 6 || month == 7 || month == 8)
        printf("\nSummer is the season");
    if (month == 9 || month == 10 || month == 11)
        printf("\nFall is the season");
    return;
}
```

- **The Function whichweek:**

```
/*  Function whichweek
 *  Input:
 *      day - a number specifying a day within a month
 *  Process:
 *      determines which week of the month day is in.
 *      assumes each week has 7 days.
 *  Output:
 *      prints the week within the month
 */
void whichweek(int day)
{
    if (day <= 7)
        printf("\n1 is the week number for day %d",day);
    if (day >= 8 && day <= 14)
        printf("\n2 is the week number for day %d",day);
    if (day >= 15 && day <= 21)
        printf("\n3 is the week number for day %d",day);
    if (day >= 22 && day <= 28)
        printf("\n4 is the week number for day %d",day);
    if (day >= 29)
        printf("\n5 is the week number for day %d",day);
    return;
}
```

- **Outline for the Complete Program:**

```
/* program to classify month-day combinations */
#include < stdio.h>

// Function Prototypes
void classify(int,int);
void translate(int);
void whichseason(int);
void whichweek(int);

void main()
{
   int month,day;
   char answer;
   ...
}

void classify(int month, int day)
{
   ...
}

void translate(int month)
{
      ...
}

void whichseason(int month)
{
   ...
}

void whichweek(int day)
{
   ...
}
```

Nested if Statements

- **Problem:**
  Write a C statement to do the following:
  If *condition_1* is true do *statement_1*. If *condition_1* is false and *condition_2 is true* do *statement_2*. If both *condition_1* and *condition_2* are false do *statement_3*.

- **First Solution:**

```
if (condition_1) statement_1;
if (!condition_1 && condition_2) statement_2;
if (!condition_1 && !condition_2) statement_3;
```

- **Solution Using Nested if statement:**

```
if (condition_1)
    statement_1;
else
    if (condition_2)
        statement_2;
    else
        statement_3;
```

- **Alternate Form of the Nested if Statement:**

```
if (condition_1)
    statement_1;
else if (condition_2)
    statement_2;
else
    statement_3;
```

- **Another example of a Nested if:**

```
if (condition_1)
    if (condition_2)
        statement_1;
    else statement_2;
else statement_3;
```

- **The Function whichweek using a Nested if:**

```
/* Function whichweek
 * Input:
 *     day - a number specifying a day within a month
 * Process:
 *     determines which week of the month day is in.
 *     assumes each week has 7 days.
 * Output:
 *     prints the week within the month
 */
void whichweek(int day)
{
    if (day <= 7)
        printf("\n1 is the week number for day %d",day);
    else if (day <= 14)
        printf("\n2 is the week number for day %d",day);
    else if (day <= 21)
        printf("\n3 is the week number for day %d",day);
    else if (day <= 28)
        printf("\n4 is the week number for day %d",day);
    else
        printf("\n5 is the week number for day %d",day);
    return;
}
```

- **Note: Simplest Implementation of the Function whichweek:**

```
void whichweek(int day)
{
    printf("\n%d is the week number for day %d", (day+6)/7,
        day);
    return;
}
```

Pitfalls in Nested if Statements

- **Problem:**
  Write a C statement to do the following:
  If *condition_1* is true then if *condition_2* is true do *statement_1*. If *condition_1* is false do *statement_3*.

- **Incorrect Solution:**

```
if (condition_1)
    if (condition_2)
        statement_1;
else
    statement_3;
```

  - Compiler's Interpretation is:

```
if (condition_1)
    if (condition_2)
        statement_1;
    else
        statement_3;
```

- **First Solution:** {null else clause}

```
if (condition_1)
    if (condition_2)
        statement_1;
    else;                    {null else clause}
else
    statement_3;
```

- **Second Solution:** use braces to delimit scope of if clause

```
if (condition_1) {
    if (condition_2)
        statement_1;
}
else
    statement_3;
```

switch Statement

- **General Syntax of the switch Statement:**

```
switch (selector) {
    case value_1:
        st_1;
        st_2;
        ...
        break;
    case value_2:
        st_3;
        ...
        break;
    ...
    case value_n:
        ...
        break;
}
```

- Note: *selector* is an integer expression.

- **Example:**

```
int anynumber,remainder;
...
remainder =  anynumber % 3;
switch (remainder) {
    case 0:
        printf("%d is a multiple of 3\n",anynumber);
        break;
    case 1:
        printf("%d is 1 more than a multiple of 3\n",anynumber);
        break;
    case 2:
        printf("%d is 2 more than a multiple of 3\n",anynumber);
        break;
}
```

Using a default Clause

- **Example:**

```
int number;

printf("Type in a number between 2 and 9: ");
scanf("%d",&number);
switch (number) {
    case 2:
        printf("\n%d is an even prime number\n",number);
        break;
    case 4:
    case 6:
    case 8:
        printf("\n%d is an even non-prime number\n",number);
        break;
    case 3:
    case 5:
    case 7:
        printf("\n%d is an odd prime number\n",number);
        break;
    case 9:
        printf("\n%d is an odd non-prime number\n",number);
        break;
    default:
        printf("\n%d is not in the range from 2 to 9\n",number);
        break;
}
```

- **Restrictions on the switch Statement:**
  - the selector must be an expression whose underlying data type is an integer **(int or char)** (no real or string).
  - each alternative must be a constant (no variables or expressions).

Loop Control Features

- **The continue Statement:**
  Causes a loop to skip to the update step.

```
for (i = 4; i <= 6; i++) {
    /* first part of the process */
    printf("first part with i = %d\n",i);
    if (i == 5)
        continue;
    /* second part of the process */
    printf("second part with i = %d\n",i);
}
```

- Output:

```
first part with i = 4
second part with i = 4
first part with i = 5
first part with i = 6
second part with i = 6
```

- **The break Statement:**
  Causes a loop to terminate.

```
for (i = 4; i <= 6; i++) {
    /* first part of the process */
    printf("first part with i = %d\n",i);
    if (i == 5)
        break;
    /* second part of the process */
    printf("second part with i = %d\n",i);
}
```

- Output:

```
first part with i = 4
second part with i = 4
first part with i = 5
```

Loop Control Features: Examples

- **Example:**

```
int number;

do {
    printf("\nType in a number: ");
    scanf("%d",&number);
    /* first part of the process */
    printf("first part: number = %d\n",number);
    if (number < 0)
        continue;
    /* second part of the process */
    printf("second part: number = %d\n",number);
} while (number != 0);
```

- **Example:**

```
int number;

while(1 == 1){                          //Note: you can use while(1)
    printf("\nType in a number: ");
    scanf("%d",&number);
    if (number == 0)
        break;
    /* first part of the process */
    printf("first part: number = %d\n",number);
    if (number < 0)
        continue;
    /* second part of the process */
    printf("second part: number = %d\n",number);
}
```

The exit() Function

- **The exit() Function:**
  - Calling the **exit** function causes the **entire program to terminate** with control returning to the operating system.

  - A return value can be sent to indicate whether or not the program terminated normally.

  - Typically, a return value of 0 is used to indicate normal program termination. Any other value (e.g., 1, 2, ...0 can be used to indicate abnormal terminations.

- **Example:**

```
if (month <  1 || month >  12) {
    printf("\n%d is an invalid value for month\n",month);
    printf("The program will terminate prematurely\n");
    exit(1);
}
else
    ...
```

- Note:
  To use the exit() function: #include < stdlib.h>

Precedence of Arithmetic, Assignment, Logical & Relational Operators

Precedence	Associativity
a) !, Unary Minus, Unary Plus, + + , --	right to left
b) * , /, %	left to right
c) + , -	left to right
d) < , < = , > , > =	left to right
e) = = , !=	left to right
f) &&	left to right
g) ||	left to right
h) = , + = , -= , * = , /= , %=	right to left

Boolean Expressions

- **Boolean Expressions:**
  A boolean expression is an expression that evaluates to either **true** or **false**.

- **Example:**

```
int x = 4, y = 0, z = 1;

if (x < y || z >= x && x != 0)
    printf("it is true\n");
else
    printf("it is false\n");
```

- **Example:**

```
int a = 2, b = 4, c = 7, d = 4, e = 3;

if (a * 2 > b || !(c == 6) && d - 1 <= e)
    printf("it is true\n");
else
    printf("it is false\n");
```

- **Example - Short-Circuit Evaluation:**

```
int x,y,z;

if (x != 0 && y / x > z)        //This works even if x == 0
    printf("it is true\n");
else
    printf("it is false\n");
```