Outline

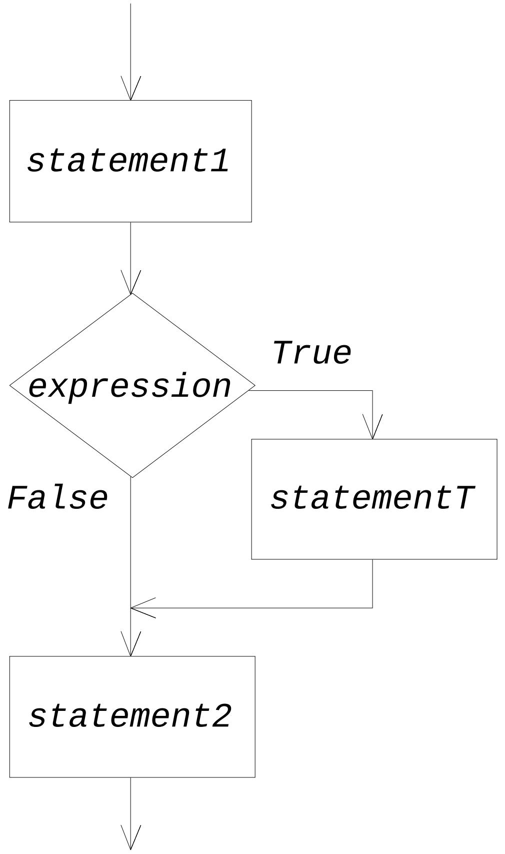
- Selection control structures
  - if
  - if-else
  - switch
- Iteration control structures
  - while
  - for
  - do-while
  - break, continue

`if` Statement

```
statement1 ;

if ( expression )
  statementT ;

statement2 ;
```



if Statement

```
if ( x < 0 ) x = -x;

if ( x == 9 )
{
  printf("Job terminates\n");
  printf("Bye bye.\n");
  exit(0);
}
```

`if` Statement

```
if ( x < 0 );
  x = 999;

if ( x = 0 ) exit(0);

if ( x > y + 5 )
{
  z = x + y;
  q = x * y;
};
```

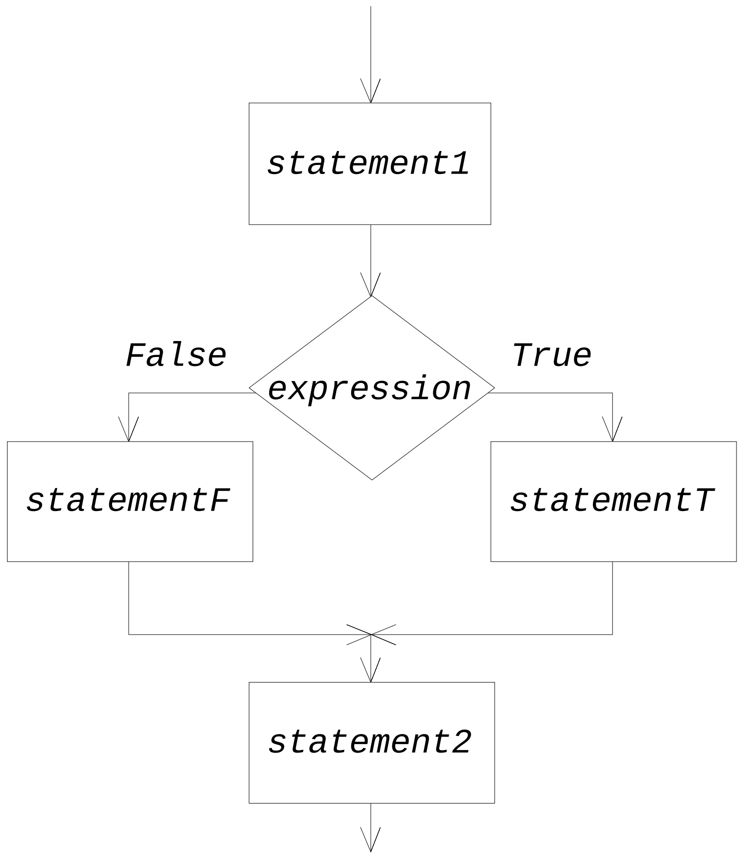
WRONG

if-else Statement

```
statement1 ;

if ( expression )
  statementT ;
else
  statementF ;

statement2 ;
```



S. Prasitjutrakul

if-else Statement

```
#include <stdio.h>
main()
{
  int    x, y;

  printf("Input an integer value for x : ");
  scanf( "%d", &x );
  printf("Input an integer value for y : ");
  scanf( "%d", &y );
  if ( x == y )
    printf("--- x is equal to y ---\n");
  else
    if ( x > y )
      printf("--- x is greater than y ---\n");
    else
      printf("--- x is less tham y ---\n");
 }
```

`if-else` Statement

```
if ( code == 1 )
  printf("Student member\n");
else
  if ( code == 2 )
    printf("Regular member\n");
  else
    if ( code == 3 )
      printf("Senior member\n");
    else
      if ( code == 4 )
        printf("Fellow\n");
      else
        printf("Error : Illegal member code\n");
```

if-else Statement

```
if ( code == 1 )
  printf("Student member\n");
else
if ( code == 2 )
  printf("Regular member\n");
else
if ( code == 3 )
  printf("Senior member\n");
else
if ( code == 4 )
  printf("Fellow\n");
else
  printf("Error : Illegal member code\n");
```

`if-else` Statement

```
if ( code == 1 )
  printf("Student member\n");
else if ( code == 2 )
  printf("Regular member\n");
else if ( code == 3 )
  printf("Senior member\n");
else if ( code == 4 )
  printf("Fellow\n");
else
  printf("Error : Illegal member code\n");
```

Do & Don't

- Don't use the "not equal to" operator (`!=`) in an if statement containing an `else`.

```
if ( code != 1 )
  printf("Not student member\n");
else
  printf("Student member\n");

if ( code == 1 )
  printf("Student member\n");
else
  printf("Not student member\n");
```

Do & Don't

- Do use `(expr == 0)` instead of `(!expr)`.
- Do use the logical operators && and || instead of nesting if statements.
- Don't confuse the assignment operator (=) with the equal to operator (==).

`switch-case` Statement

```
switch ( code ) {
  case 1 :
       printf("Student member\n");
       break;
  case 2 :
       printf("Regular member\n");
       break;
  case 3 :
       printf("Senior member\n");
       break;
  case 4 :
       printf("Fellow\n");
       break;
  default :
       printf("Error : Illegal member code\n");
       break;
}
```

switch-case Statement

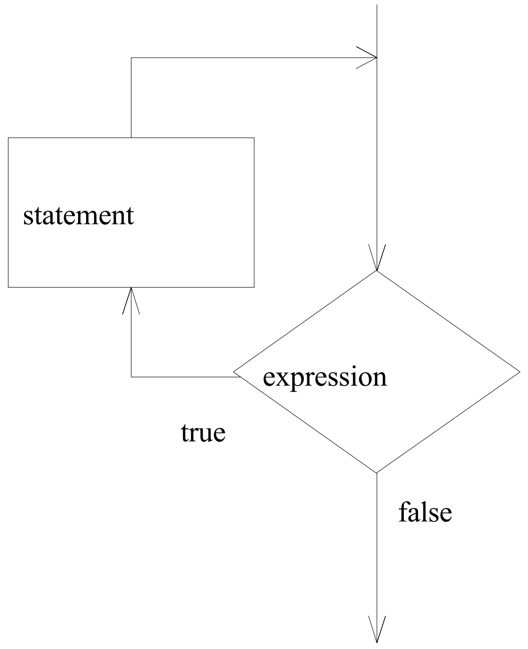
```
 switch (c) {
   case '0' :
   case '1' :
   case '2' :
   case '3' :
   case '4' :
   case '5' :
   case '6' :
   case '7' :
   case '8' :
   case '9' :
        nDigit[ c-'0' ]++;
        break;
   default :
        nOther++;
        break;
}
```

Do & Don't

- Do use a default case in a `switch` statement even if you think you have covered all the possibles cases.
- Do use a `switch` statement instead of an `if` statement if there are more than two conditions being evaluated for the same var.
- Do line up your `case` statements for readability.
- Don't forget to use `break` statements if your `switch` statements need them.

`while` Statement

```
while ( expression )
  statement
```



`while` Statement

```
while ( (c = getchar()) != EOF )
  switch (c) {
    case '0' :
    case '1' :
    case '2' :
    case '3' :
    case '4' :
    case '5' :
    case '6' :
    case '7' :
    case '8' :
    case '9' :
         nDigit[ c-'0' ]++;
         break;
    default :
         nOther++;
         break;
 }
```

while Statement

```
csh> cp  sourceFile     destinationFile
```

```
while ( (c = getchar()) == ' ' ||
         c == '\n' ||
         c == '\t' )
  ;
```

`for` Statement

- expr1 and expr3 are assignments
- expr2 is relational expression
- Any of the three parts can be omitted (the semicolons must remain)
- If the expr2 is not present, it is taken as permanently true.

```
for ( expr1; expr2; expr3 )
   statement
```

```
expr1;
while ( expr2 ) {
  statement
  expr3;
}
```

`for` Statement

```
for (i = 0; i < N; i++ ) {
  ...
}

for (j = N; j > 0; j-- ) {
  ...
}

for (i = 0; i < N; i++) {
  for (j = 0; j < i; j++) {
    ...
  }
}
```

for : atoi Function

```
int atoi( char s[] )
{
  int    i, n, sign;

  for (i=0; s[i]==' ' || s[i]=='\n' || s[i]=='\t'; i++ )
    ;
  sign = 1
  if (s[i] == '+' || s[i] =='-')
    sign = (s[s++]=='+') ? 1 : -1;
  for (n=0; s[i]>='0' && s[i] <= '9'; i++)
    n = 10*n + (s[i]-'0');
  return( sign*n );
}
```

for Statement

```
/* two infinite loops */
for ( ;; ) {
  ...
}
while ( 1 ) {
  ...
}
```

```
for ( n=0; n!=99; scanf("%d", &n) ) ;

for (n=0; n!=99; )
  scanf( "%d", &n );

scanf( "%d", &n );
for ( ;n!=99; scanf("%d", &n ) ) ;

scanf( "%d", &n );
for ( ;; ) {
  if ( n == 99 ) break;
  scanf( "%d", &n );
}
```

for : reverse Function

```
reverse( char s[] )
{
  int     c, i, j;

  for ( i=0, j=strlen(s)-1; i<j; i++, j++ ) {
    c    = s[i];
    s[i] = s[j];
    s[j] = c;
  }
}
```

H	E	L	L	O	'\0'

O	L	L	E	H	'\0'

S. Prasitjutrakul

Comma Operator

- A pair of expressions separated by a comma is evaluated left to right.
- The type and value of the result are the type and value of the right operand.

```
 a = ( i++, --j );

i++;
a = --j;
```

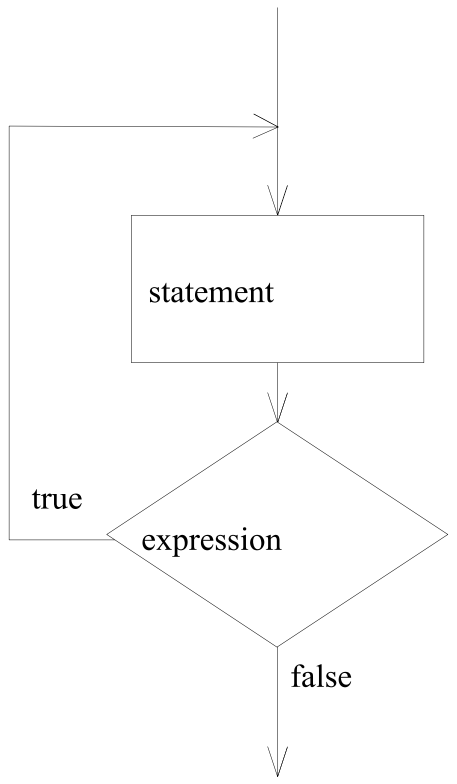
The commas that separate function arguments, variables in declarations are not comma operators, do not guarantee left to right evaluatons.

Do & Don't

- Don't put too much processing in the `for`. It is often clearer to put some action into the body of the loop.
- Do remember the semicolon if you use a `for` with a null statement.

`do-while` Statement

```
do
  statement
while ( expression );
```



do-while Statement

```
int GetMenuChoice( void )
{
  int     selection = 0;

  do {
    ClearScreen();
    printf("1 - Add a Record\n");
    printf("2 - Change a Record\n");
    printf("3 - Delete a Record\n");
    printf("4 - Quit\n");
    printf("\nEnter a selection : ");
    scanf( "%d", &selection );
  } while (selection < 1 || selection > 4);

  return selection;
}
```

do-while : itoa Function

```c
void itoa( int n, char s[] )
{
  int     i, sign;

  sign = (n < 0) ? -1 : 1;
  if ( n < 0 ) n = -n;
  i = 0;
  do {
    s[i++] = (n % 10) + '0';
    n /= 10;
  } while ( n > 0 );
  if ( sign < 0 ) s[i++] = '-';
  s[i] = '\0';
  reverse( s );
}
```

Nested Loops

```
for (i=1; i<N; i++) {
  ...
  for (j=i; j>0; j--) {
     ...
  }
  ...
}
```

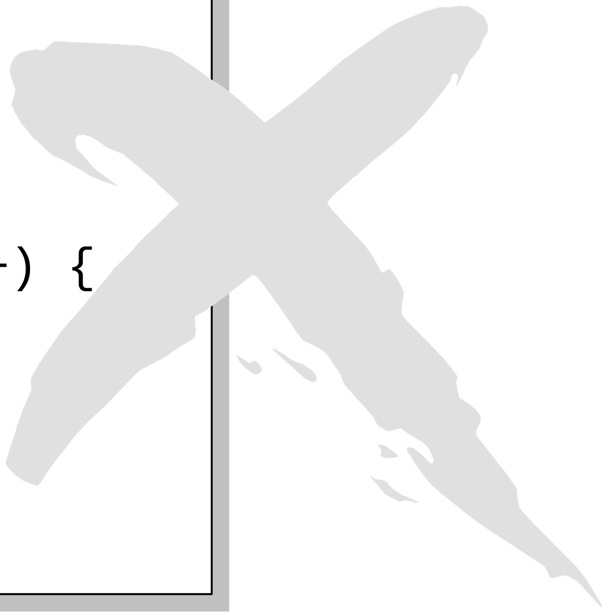
```
while ( i != 0 ) {
  ...
  while ( i+5 < j ) {
    ...
  }
  ...
}
```

```
do {
  ...
  while ( i > 0 ) {
    ...
    for (i=0;i<j;i++) {
      ...
    }
  }
} while ( i++ > 0 );
```

Each inner loop must be enclosed completely in the outer loop

Nested Loops

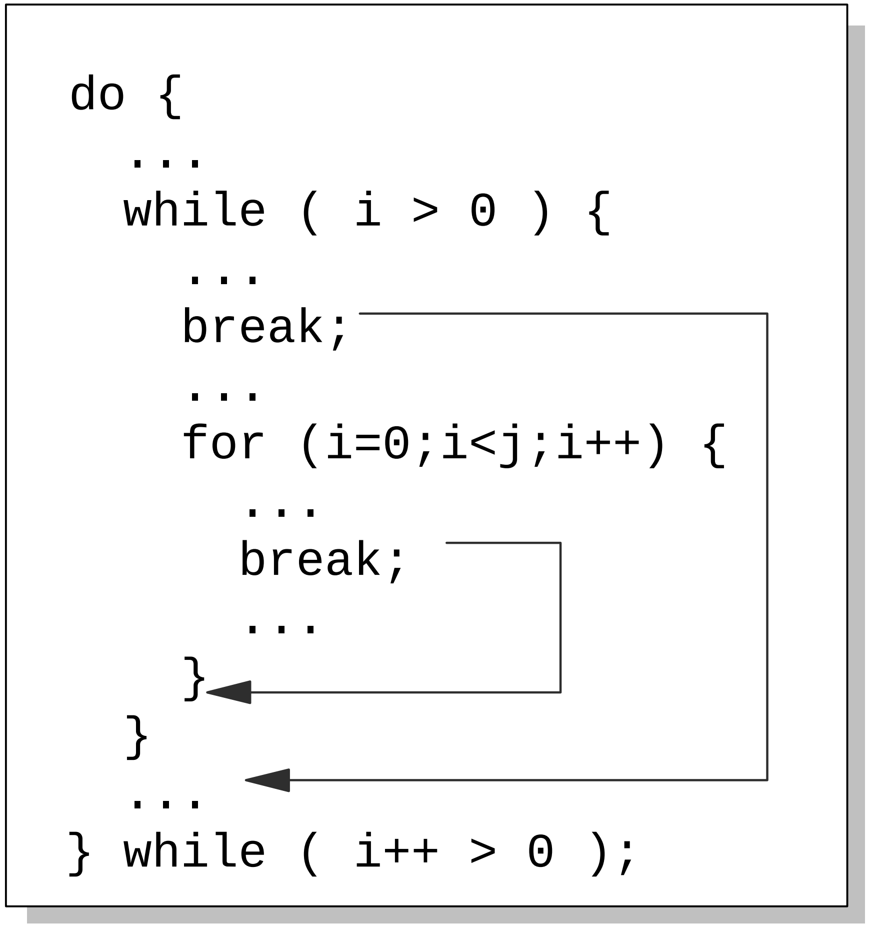
```
do {
  ...
  while ( i > 0 ) {
    ...
    for (i=0;i<j;i++) {
      ...
    }
} while ( i++ > 0 );
  }
```



Good indenting style makes code with nested loops easier to read.
Each level of loop should be one step farther that the last level.

`break` Statement

- provide an early exit from `for`, `while`, `do`, and `switch`.
- cause the control to skip past the current loop.



```
do {
   ...
   while ( i > 0 ) {
      ...
      break;
      ...
      for (i=0;i<j;i++) {
         ...
         break;
         ...
      }
   }
   ...
} while ( i++ > 0 );
```

S. Prasitjutrakul

Remove Trailing Blanks and Tabs

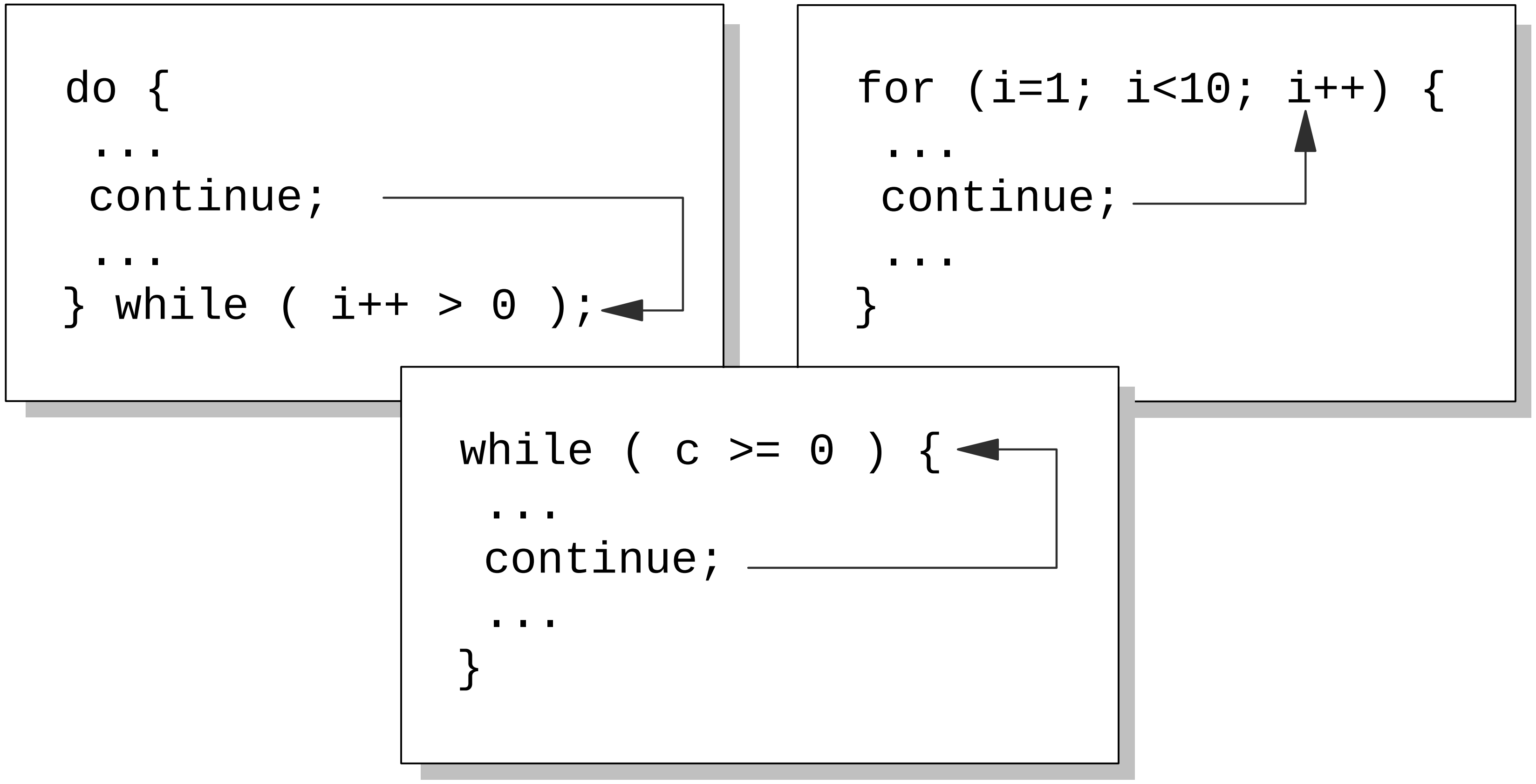
```
main()
{
  int     n;
  char    line[100];

  while( (n=getline(line, 100)) > 0) {
    while (--n >= 0)
      if ( line[n] != ' ' &&
           line[n] != '\t' &&
           line[n] != '\n')
        break;
    line[n+1] = '\0'
    printf("%s\n", line);
  }
}
```

`getline` gets a line of input and returns the length of the line

`continue` Statement

- Can be placed only in the body of a loop.
- cause the next iteration of the enclosing loop to begin.



```
do {
 ...
 continue;
 ...
} while ( i++ > 0 );
```

```
for (i=1; i<10; i++) {
 ...
 continue;
 ...
}
```

```
while ( c >= 0 ) {
 ...
 continue;
 ...
}
```

S. Prasitjutrakul

`continue` Statement

```
for (i=0; i<N, i++) {
  if ( a[i] < 0 ) continue;
  ...
}

for (i=0; i<N; i++) {
  if ( a[i] >= 0 ) {
    ...
  }
}
```