中国民航大学
Civil Aviation University of China

Computer Programming

Sino-European Institute of Aviation Engineering

Module 3-3
Control Flow-Loop

Outline

- Program Loop
- The While Loop
- The Do While Loop
- The For Loop
- Break and Continue
- Structured Programming

Program Loop

- Looping: doing one thing over and over
- Program Loop: a set of statements that is executed repetitively for a number of times

Simple example: displaying a message 100 times:

```
printf(“Hello!\n”);
printf(“Hello!\n”);
…
printf(“Hello!\n”);
printf(“Hello!\n”);
```

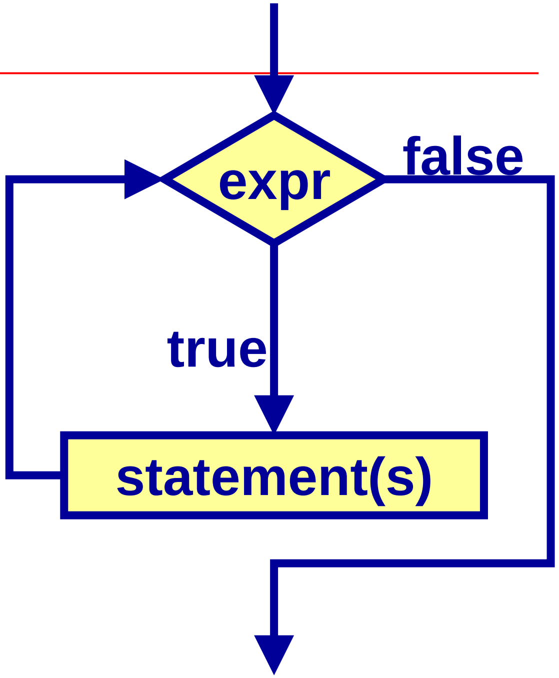
```
Repeat 100 times
printf(“Hello!\n”);
```

Program Loop

- Loop
  - Group of instructions computer executes repeatedly while some condition remains true
  - C provides flexible ways of deciding how many times to loop, or when to exit a loop.

The While Loop

```
while (exp)
   {
      statement(s);
   }
```



- The statements are executed as long as the condition is true.
- When the condition is no longer true, the loop is stopped.

The While Loop

Example : Calculation of simple interest for 3 sets of p, n and r

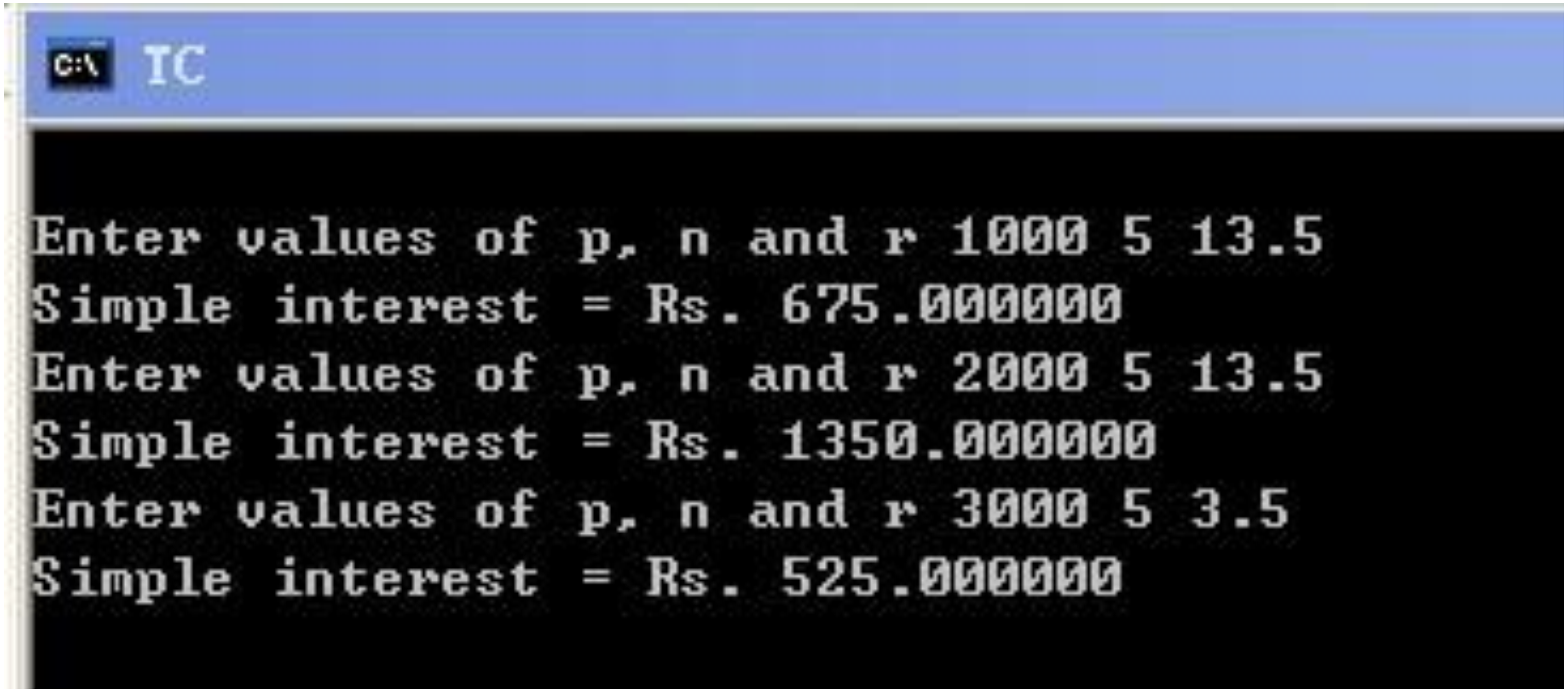
```
#include <stdio.h>
void main( )
{
    int   p, n, count ;
    float   r, si ;

    count = 1 ;
    while ( count <= 3 )
    {
           printf ( "\nEnter values of p, n and r " ) ;
           scanf ( "%d %d %f", &p, &n, &r ) ;
           si = p * n * r / 100 ;
           printf ( "Simple interest = Rs. %f", si ) ;

         count = count + 1 ;
    }
}
```

The While Loop

Example : Calculation of simple interest for 3 sets of p, n and r



```
Enter values of p, n and r 1000 5 13.5
Simple interest = Rs. 675.000000
Enter values of p, n and r 2000 5 13.5
Simple interest = Rs. 1350.000000
Enter values of p, n and r 3000 5 3.5
Simple interest = Rs. 525.000000
```

The While Loop

Example : factorial

```
int main()
{
    int i = 0, n = 0, fact = 1;

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

    i=1;        /* this is the counter */
    while (i <= n)
    {
        fact *= i;
        ++i;
    }

    printf("the factorial is %d\n", fact);
    return 0;
}
```

Initialize

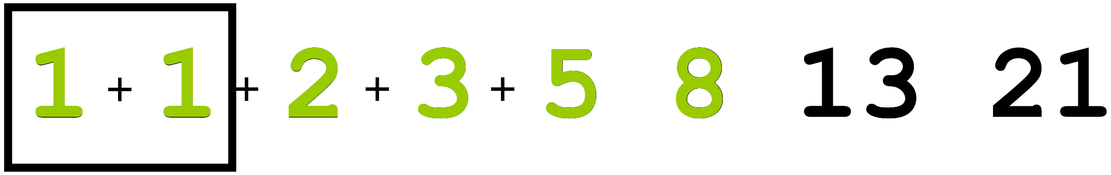
Test

Increment

The While Loop

Example - factorial

- **1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …**
- **Notice that we only need two elements in order to calculate the next one.**



1 + 1 + 2 + 3 + 5 8 13 21

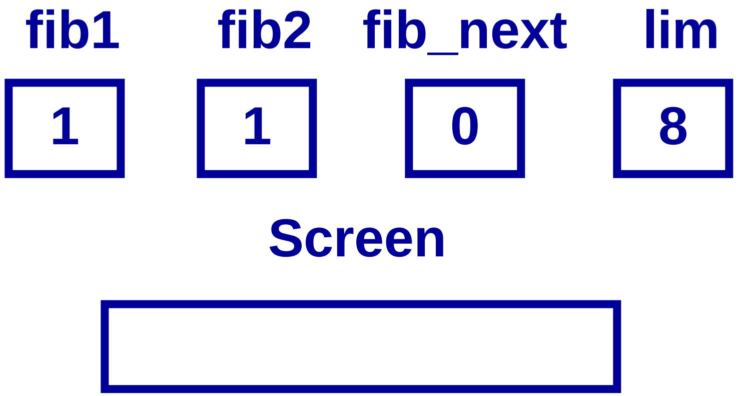
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
         printf("%d ", fib2);
         fib_next = fib1 + fib2;
         fib1 = fib2;
         fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
1	1	0	8

Screen

The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```

fib1	fib2	fib_next	lim
1	1	0	8

Screen

1

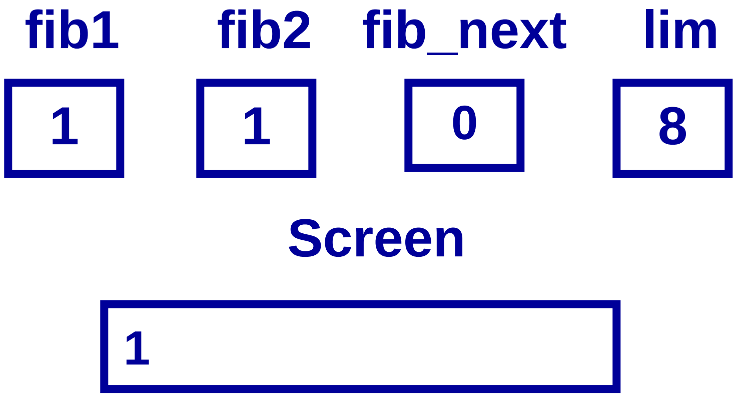
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



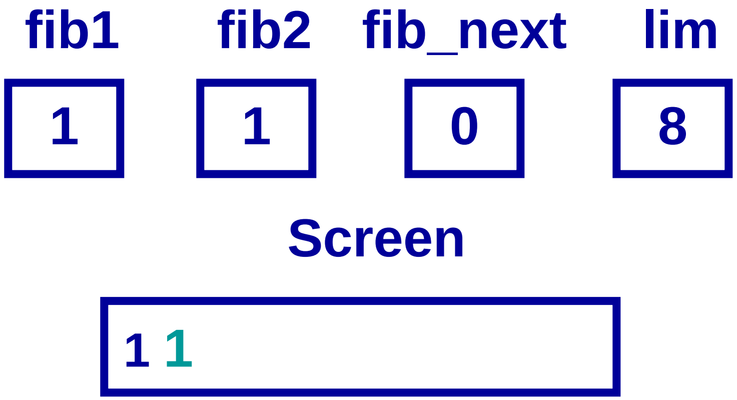
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
1	1	0	8

Screen

1 1

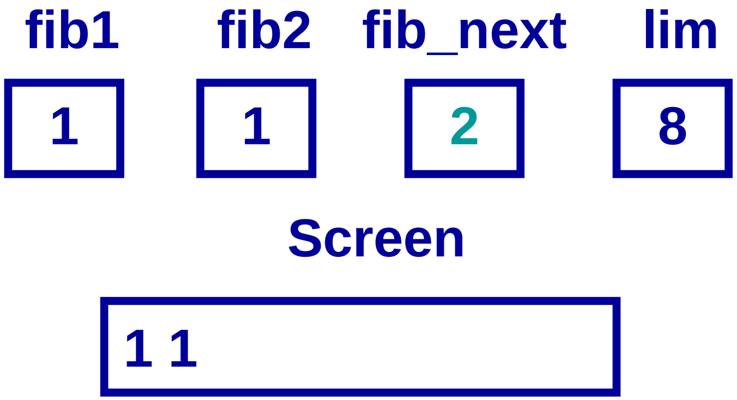
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
1	1	2	8

Screen

```
1 1
```

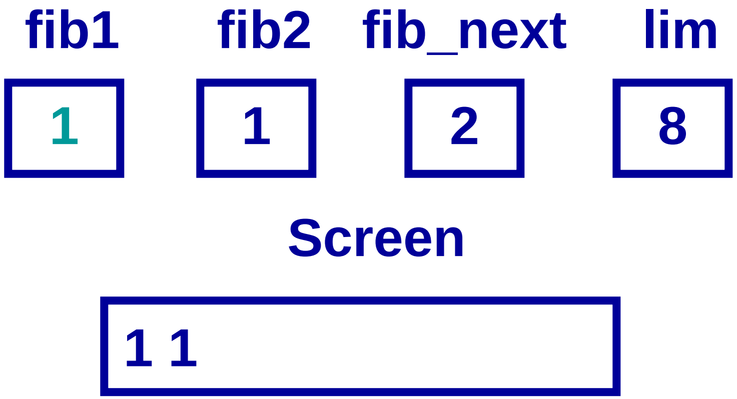
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
1	1	2	8

Screen

```
1 1
```

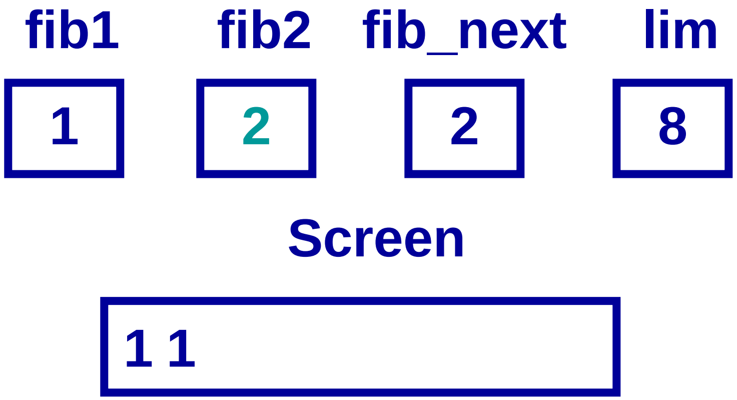
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
1	2	2	8

Screen

1 1

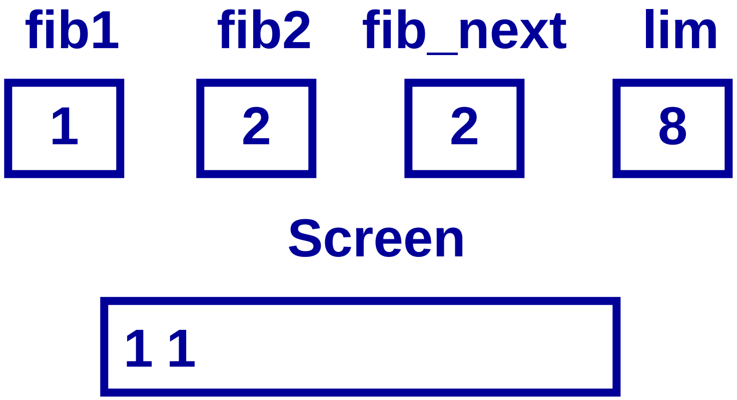
The While Loop

```
int  fib1 = 1,
     fib2 = 1,
     fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
          printf("%d ", fib2);
          fib_next = fib1 + fib2;
          fib1 = fib2;
          fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
1	2	2	8

Screen

1 1

The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```

fib1	fib2	fib_next	lim
1	2	2	8

Screen

```
1 1 2
```

The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```

fib1	fib2	fib_next	lim
1	2	3	8

Screen

```
1 1 2
```

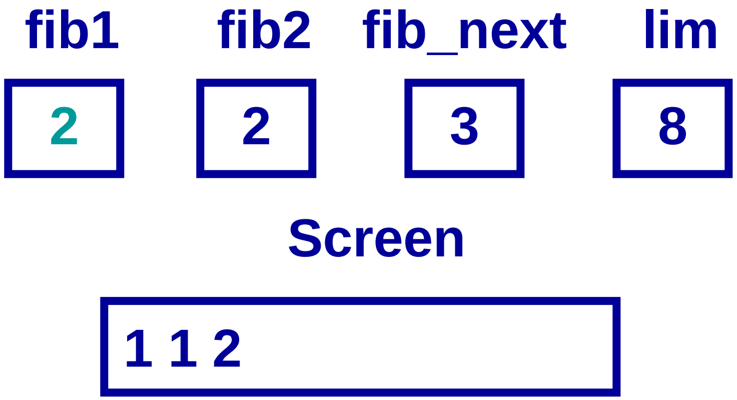
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1 fib2 fib_next lim

2 2 3 8

Screen

1 1 2

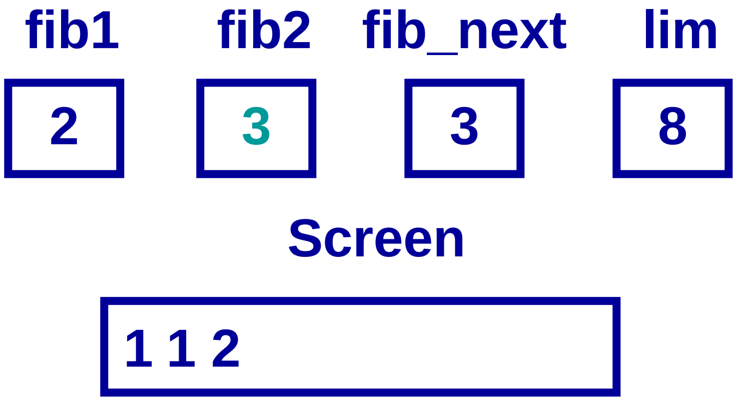
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
2	3	3	8

Screen

1 1 2

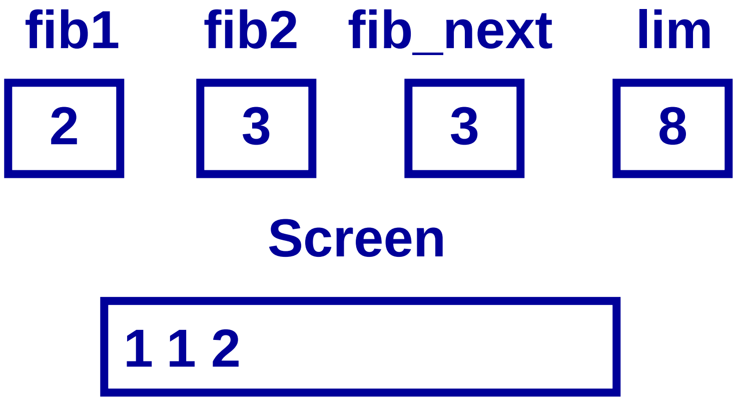
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
2	3	3	8

Screen

1 1 2

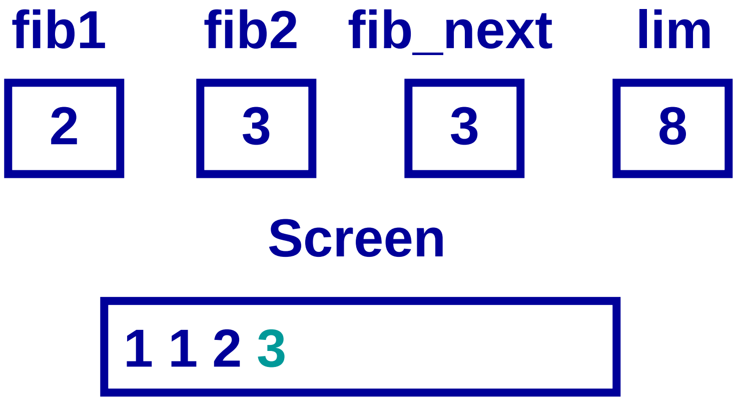
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
2	3	3	8

Screen

```
1 1 2 3
```

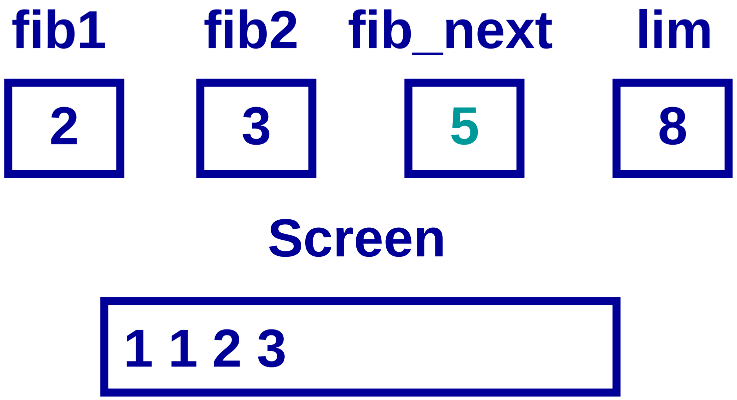
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1 fib2 fib_next lim

2 3 5 8

Screen

1 1 2 3

The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```

fib1	fib2	fib_next	lim
3	3	5	8

Screen

```
1 1 2 3
```

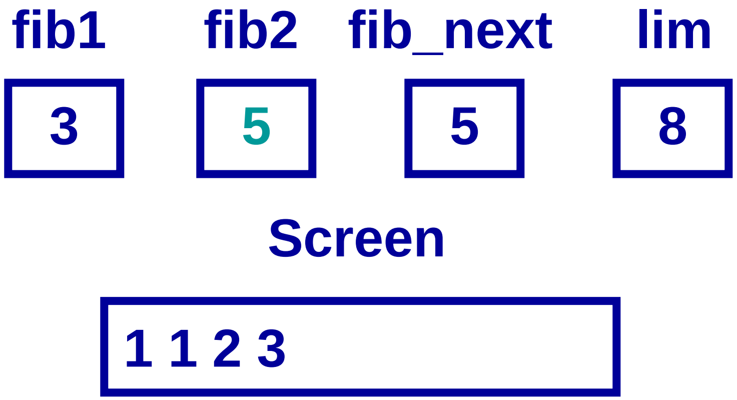
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
3	5	5	8

Screen

1 1 2 3

The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```

fib1	fib2	fib_next	lim
3	5	5	8

Screen

1 1 2 3

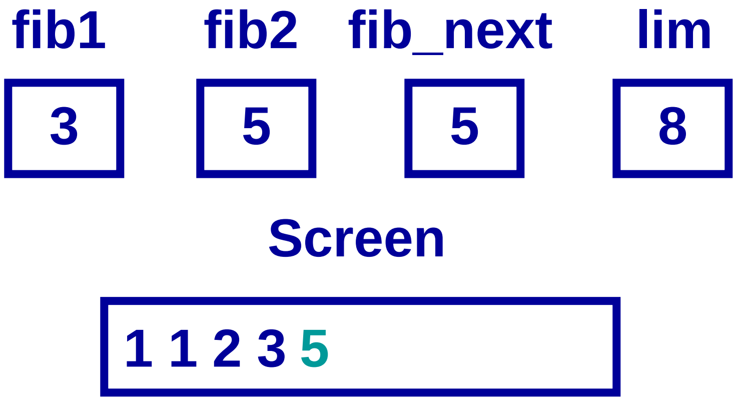
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
3	5	5	8

Screen

1 1 2 3 5

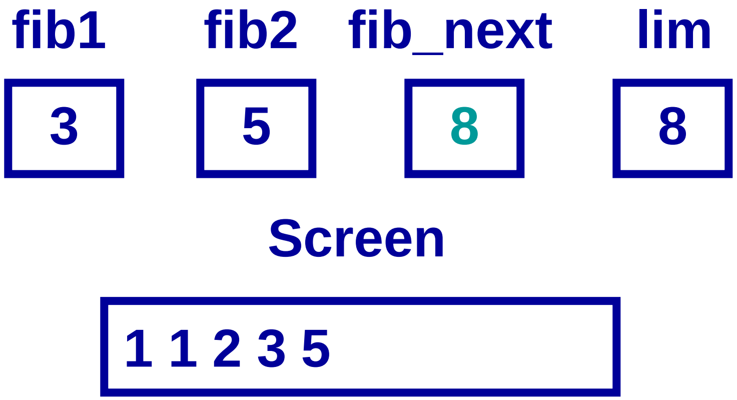
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
         printf("%d ", fib2);
         fib_next = fib1 + fib2;
         fib1 = fib2;
         fib2 = fib_next;
}

printf("\n");
```



fib1 fib2 fib_next lim

3 5 8 8

Screen

1 1 2 3 5

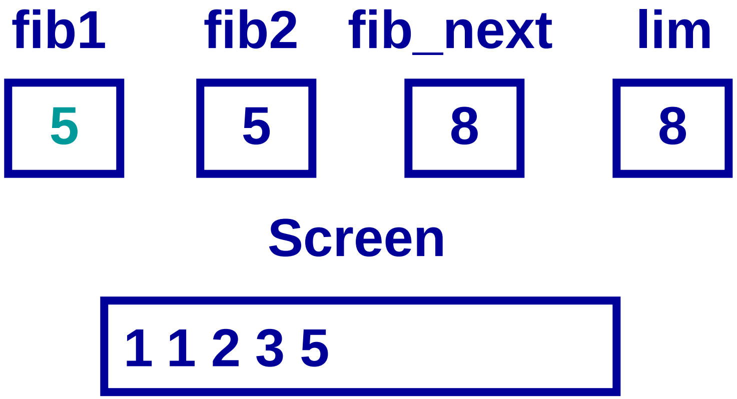
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



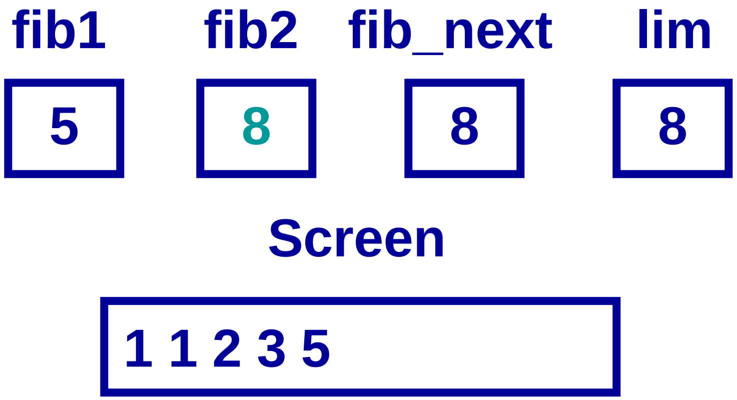
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
        printf("%d ", fib2);
        fib_next = fib1 + fib2;
        fib1 = fib2;
        fib2 = fib_next;
}

printf("\n");
```



fib1	fib2	fib_next	lim
5	8	8	8

Screen

1 1 2 3 5

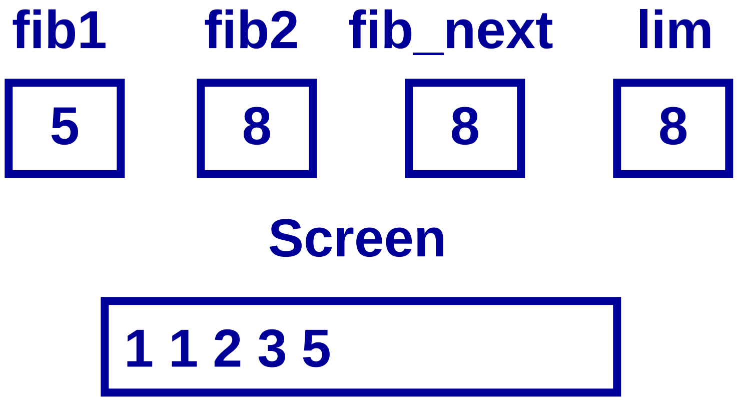
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
         printf("%d ", fib2);
         fib_next = fib1 + fib2;
         fib1 = fib2;
         fib2 = fib_next;
}

printf("\n");
```



fib1 fib2 fib_next lim

5 8 8 8

Screen

1 1 2 3 5

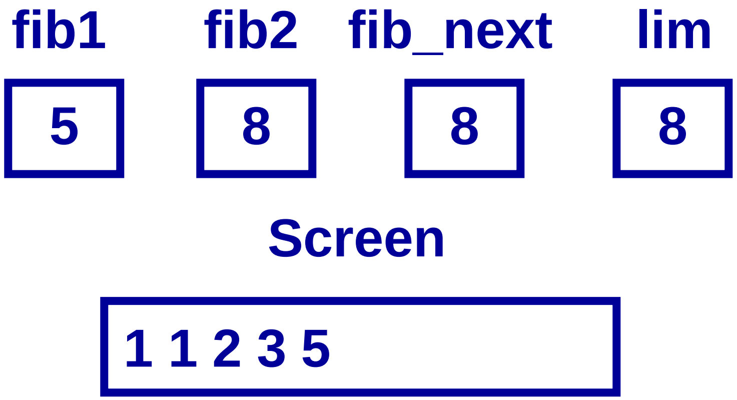
The While Loop

```
int fib1 = 1,
    fib2 = 1,
    fib_next = 0;

printf("%d ", fib1);

while (fib2 < lim)
{
          printf("%d ", fib2);
          fib_next = fib1 + fib2;
          fib1 = fib2;
          fib2 = fib_next;
}

printf("\n");
```



fib1 fib2 fib_next lim

5 8 8 8

Screen

1 1 2 3 5

The While Loop

- Counter-controlled repetition
  - Definite repetition: know how many times loop will execute
  - Control variable used to count repetitions
- Sentinel-controlled repetition
  - Indefinite repetition
  - Used when number of repetitions not known
  - Sentinel value indicates "end of data"

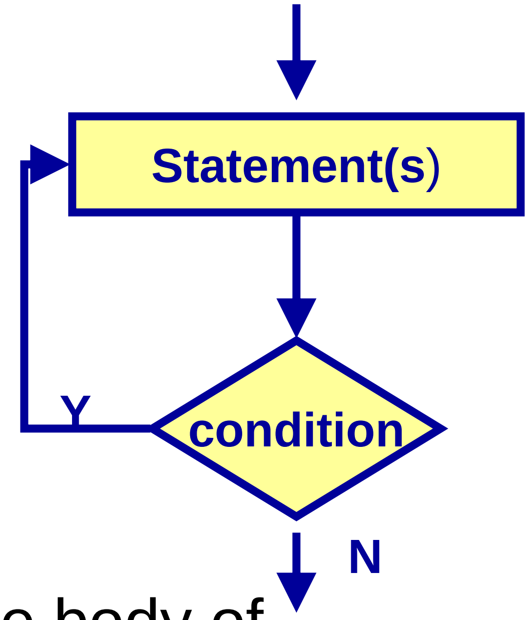
The While Loop

- Counter-controlled repetition requires
  - The name of a control variable (or loop counter)
  - The initial value of the control variable
  - A condition that tests for the final value of the control variable (i.e., whether looping should continue)
  - An increment (or decrement) by which the control variable is modified each time through the loop

The Do…While Loop

- Format:

```
do {
  statement;
} while ( condition );
```



- Similar to the while structure
- Condition for repetition tested after the body of the loop is performed
- All actions are performed at least once

The Do…While Loop

- Example: Prints the integers from 1 to 10

```
#include <stdio.h>
void main()
{
   int counter = 1;
   do {
     printf( "%d  ", counter );
   } while (++counter <= 10);
}
```

The Do…While Loop

```
#include <stdio.h>
int main()
{  int count, number;
     count = 0;
     printf(“Enter a number: ");
     scanf ("%d", &number) ;
     if (number < 0)   number = -number;
     do {
           number = number / 10;
            count ++;
     } while (number != 0);
     printf("It contains %d digits.\n", count);
    return 0;
}
```

```
Enter a number: 12534
It contains 5 digits.
```

```
Enter a number: -99
It contains 2 digits.
```

```
Enter a number: 0
It contains 1 digits.
```

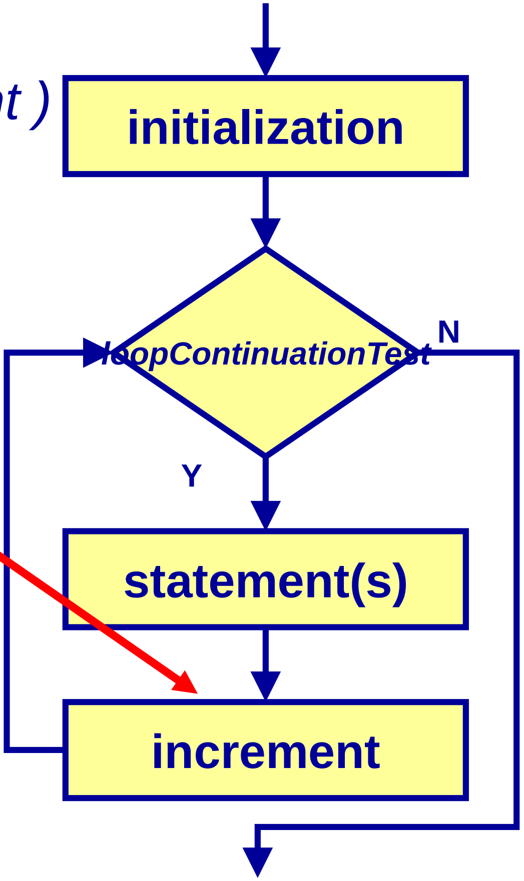
```
while (number != 0) {
      number = number / 10;
      count ++;
}
```

The For Loop

- Format

```
for ( initialization; loopContinuationTest; increment )
    statement
```

No semicolon (;) after last expression



E.g Prints the integers from one to ten

```
for( int counter = 1; counter <= 10;counter++ )
    printf( "%d|n", counter );
```

The For Loop

- For loops can usually be rewritten as while loops:

```
initialization;
while ( loopContinuationTest )
{
   statement;
   increment;
}
```

for (;   ;   ) = while (1)

for(;exp_2;)=while(exp_2)

for(exp_1; ;exp_3)=exp_1; while(1) {exp_3;}

- Initialization and increment
  - Can be comma-separated lists
  - Example:

```
for (int i = 0, j = 0;  j + i <= 10; j++, i++)
  printf( "%d\n", j + i );
```

The For Loop

```
(1) i=1;
      for (  ; i<=100; i++) sum+=i;
(2) for (i=1;  ; i++)
        { …   if(i>100)…
           …}
(3) for (i=1; i<=100; )
        {…i++; … }
(4) i=1;
      for (  ;  ;  )
        { … if(i>100) …
          i++; … }
```

The For Loop

- Arithmetic expressions
  - Initialization, loop-continuation, and increment can contain arithmetic expressions.

E.g If x equals 2 and y equals 10:

*for (j = x; j <= 4 * x * y; j += y / x)*

⇕ *Equal to*

for (j = 2; j <= 80; j += 5)

The For Loop

- Calculate $S = \sum_{k=1}^{100} k$

```
#include <stdio.h>
void main( )
{   int k,s;
    s=0;

for(k=1;k<=100;k++)
        s=s+k;
    printf("s=%d",s);
}
```

```
#include <stdio.h>
void main( )
{   int k,s;
    s=0;k=1;
    do
    {  s= s+k;
       k++;
    }while(k<=100);
    printf("s=%d",s);
}
```

```
#include <stdio.h>
void main( )
{   int k,s;
    s=0;k=1;
    while(k<=100)
    { s=s+k;
      k++;
    }
    printf("s=%d",s);
}
```

Nesting of Loops

- fors or whiles statements can be nested

```
(1)  while( )
       {…
          while( )
          {…}
       }
```

```
(2) do
     {…
        do
        {… }
              while( );
     } while( );
```

```
(3) for(;;)
     {
        for(;;)
        {… }
     }
```

```
(4)  while( )
      {…
        do{…}
           while( );
         {…}
       }
```

```
(5) for(;;)
     {…
          while( )
              {  }
       …
        }
```

```
(6) do
     {…
       for(;;){ }
         …
     }
        while( );
```

Nesting of Loops

Example: Demonstration of nested loops

```
1 × 1=1   1 × 2=2    1 × 3=3   1 × 4=4    1 × 5=5
2 × 1=2   2 × 2=4    2 × 3=6   2 × 4=8
3 × 1=3   3 × 2=6    3 × 3=9
…
5 × 1=5
```

```
for (i=1;i<=3;i++)
{      for(j=1;j<=5;j++)
       {  printf(“%d*%d=%3d”,i,j,i*j);
       }
       printf(“\n”);
}
```

Nesting of Loops

- Nesting of loops

write a program to produce the following output.

```
******
******
******
******
```

```
#include <stdio.h>
void main()
{
    int i,j;
    for(i=1;i<=4;i++)
         {  for(j=1;j<=6;j++)
                 printf(“*”);
               printf(“\n”);
              }
    }
```

Nesting of Loops

- Nesting of loops

  write a program to produce the following output.

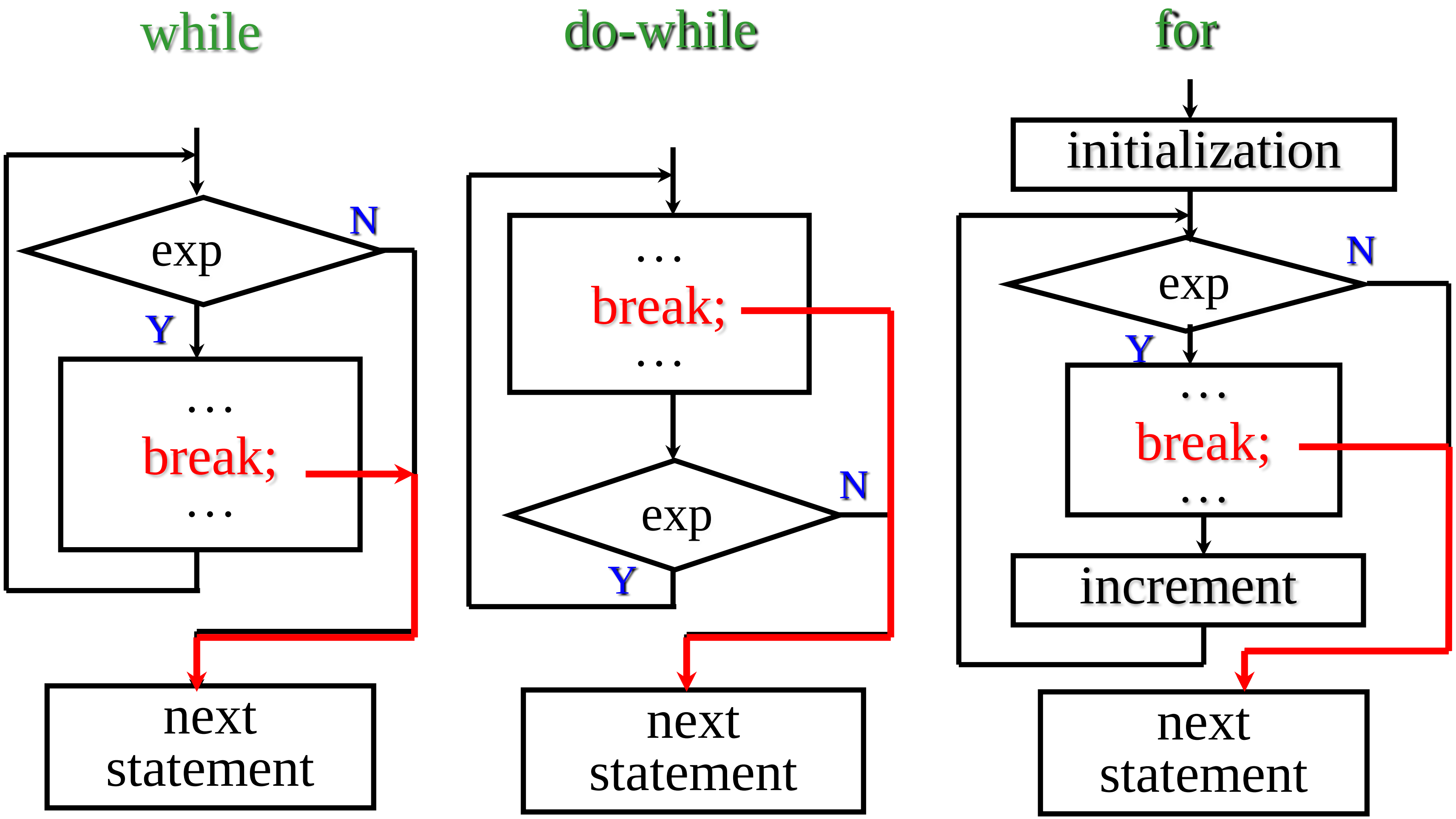
```
******
*****
****
***
**
*
```

```
#include <stdio.h>
void main()
{
   int i,j;
   for(i=1;i<=6;i++)
        {  for(j=1;j<=7-i;j++)
              printf(“*”);
           printf(“\n”);
           }
   }
```

Break Statement

- Break
  - Causes immediate exit from a while, for, do/while or switch structure
  - Program execution continues with the first statement after the structure
  - Common uses of the break statement
    - Escape early from a loop
    - Skip the remainder of a switch structure

Break Statement



Break Statement

- E.g

```
⑴ int x,n=0,s=0;
    while (n<10)
     { scanf("%d",&x);
       if (x<0) break;
       s+=x; n++;
     };
```

```
⑵ int x,n=0,s=0;
     do
      { scanf("%d",&x);
        if (x<0) break;
        s+=x; n++;
       } while (n<10);
```

```
⑶ for (n=0,s=0; n<10; n++ )
      {  scanf("%d",&x);
         if (x<0) break;
         s+=x;
       }
```

Break Statement

```
#include <stdio.h>
void main()
{  int i, m;
  printf(“Enter a number: ");
  scanf ("%d", &m);
  for (i = 2; i <= m/2; i++)
        if (m % i == 0)  break;
  if (i > m/2 )
        printf("%d is a prime number! \n", m);
  else
        printf("No!\n");
}
```

```
Enter a number: 9
No
```

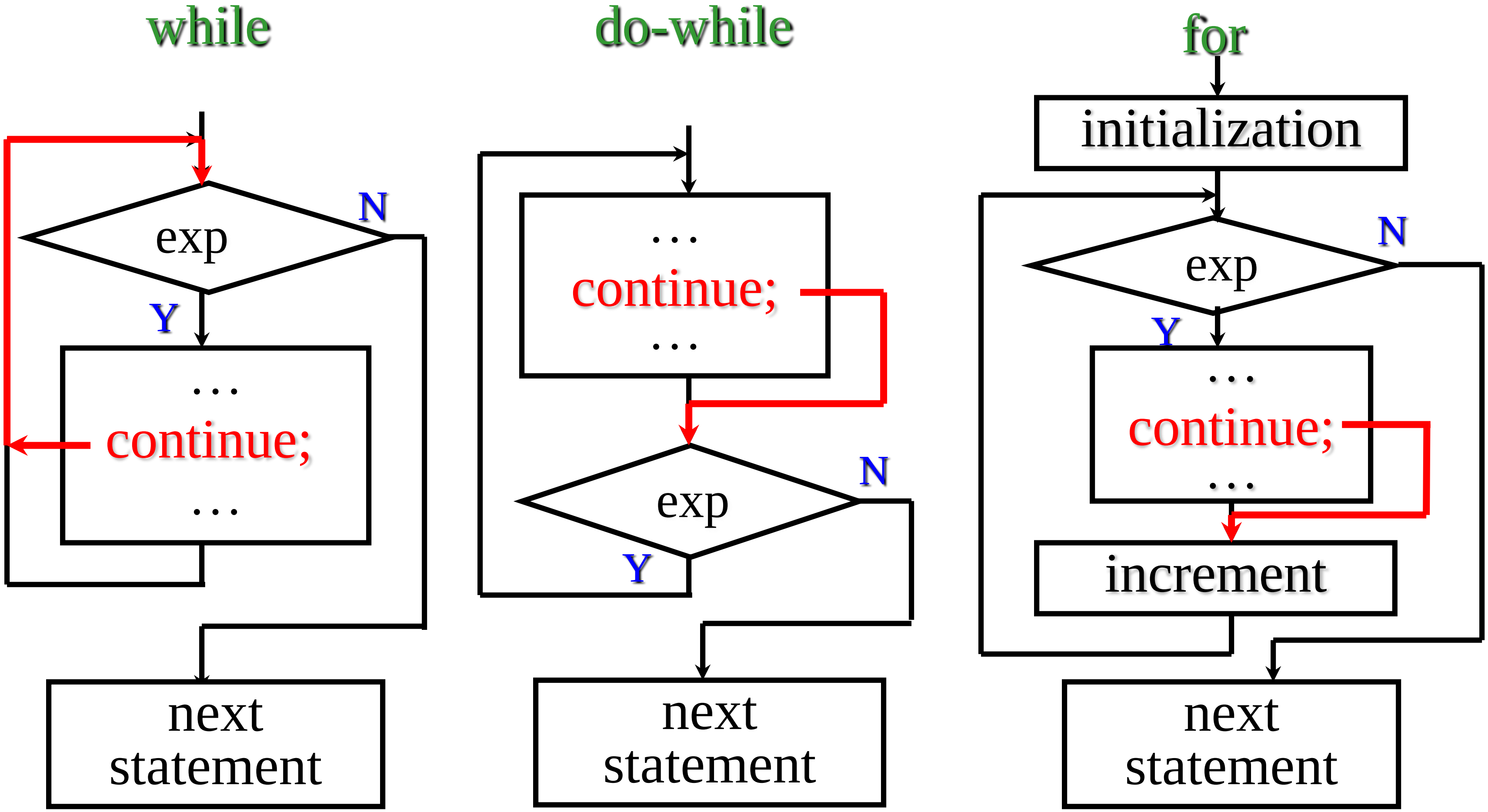
```
Enter a number: 11
11 is a prime number!
```

```
for (i = 2; i <= m/2; i++)
   if (m % i == 0) printf("No!\n");
   else printf("%d is a prime number! \n", m);
```

Continue Statement

- Continue
  - Skips the remaining statements in the body of a while, for or do/while structure
    - Proceeds with the next iteration of the loop
  - while and do/while
    - Loop-continuation test is evaluated immediately after the continue statement is executed
  - for
    - Increment expression is executed, then the loop-continuation test is evaluated

Continue Statement



Continue Statement

- E.g

```
(1) int x,n=0,s=0;
     while (n<10)
      { scanf("%d",&x);
        if (x<0) continue;
        s+=x; n++;
       };
```

```
(2) int x,n=0,s=0;
     do
      { scanf("%d",&x);
        if (x<0) continue;
        s+=x; n++;
       } while (n<10);
```

```
(3) for (n=0,s=0; n<10; n++)
       { scanf("%d",&x);
         if (x<0) continue;
         s+=x;
        }
```

Continue Statement

```
void main( )
{ int n,j=0;
   for(n=100;n<=200;n++)
    { if (n%7!=0)
           continue;
        printf("%6d",n);
        j++;
        if (j%10==0)
           printf("\n");
       }
printf(" \n j=%d\n",j);
 }
```

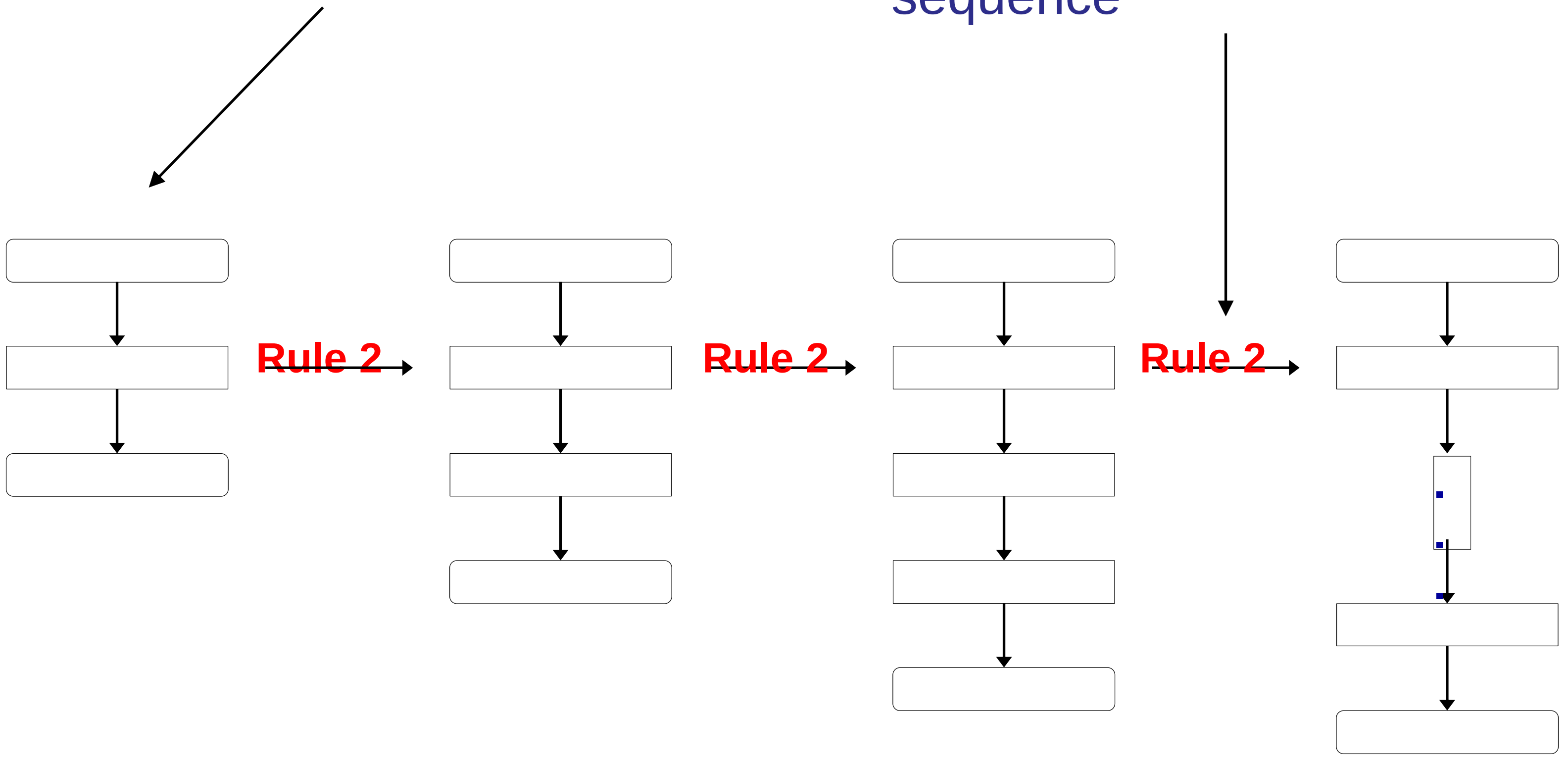
Structured Programming

- Why?
  - Easier than unstructured programs to understand, test, debug and, modify programs
- Rules for structured programming
  - Rules developed by programming community
  - Only single-entry/single-exit control structures are used
  - Rules:
    - Begin with the “simplest flowchart”
    - Any rectangle (action) can be replaced by two rectangles (actions) in sequence
    - Any rectangle (action) can be replaced by any control structure (sequence, if, if/else, switch, while, do/while or for)
    - Rules 2 and 3 can be applied in any order and multiple times

Structured Programming

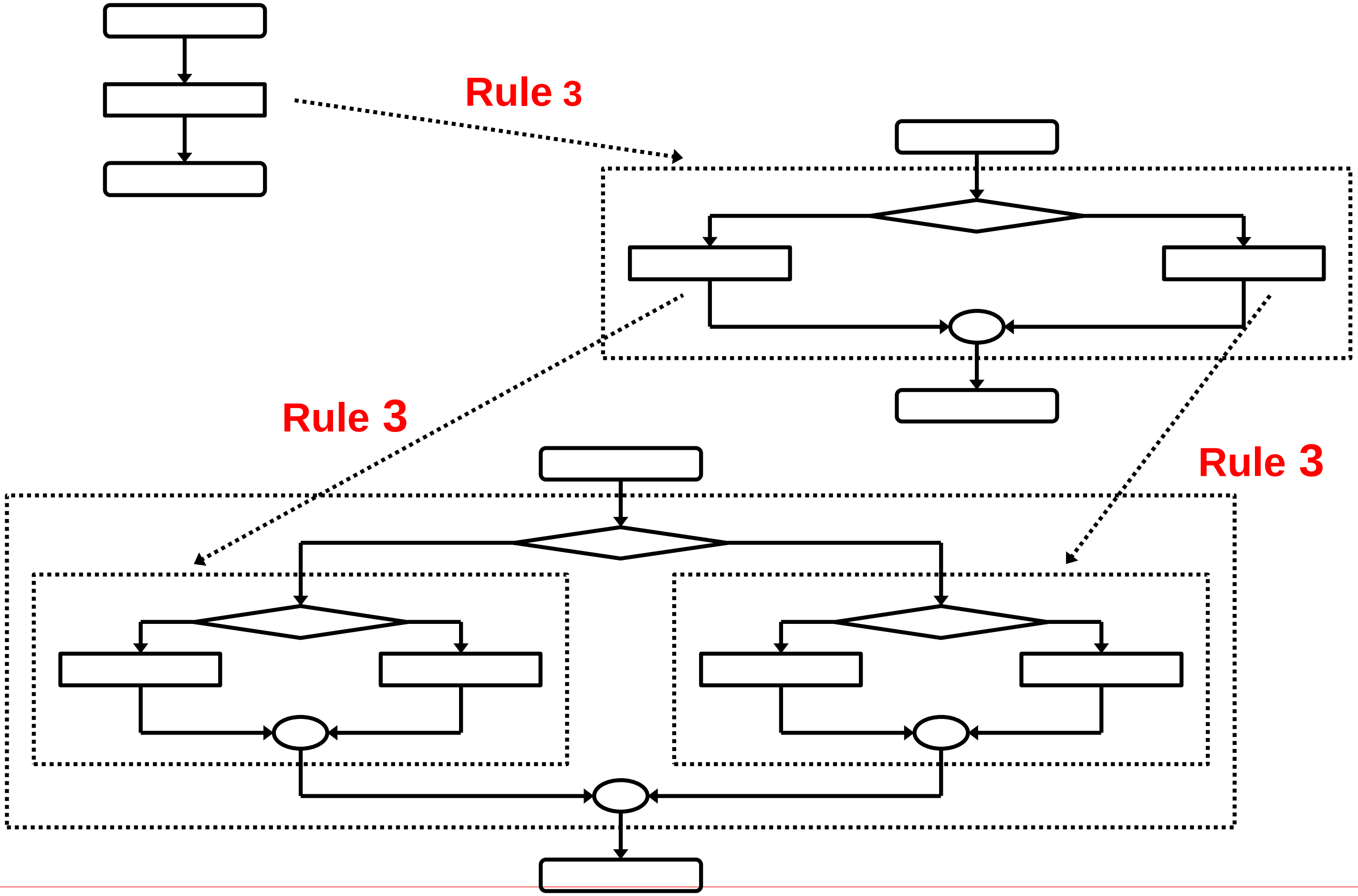
Rule 1 - Begin with the simplest flowchart

Rule 2 - Any rectangle can be replaced by two rectangles in sequence



Structured Programming

Rule 3 - Replace any rectangle with a control structure



Structured Programming

- All programs can be broken down into 3 controls
  - Sequence – handled automatically by compiler
  - Selection – if, if/else or switch
  - Repetition – while, do/while or for
- Can only be combined in two ways
  - Nesting (rule 3)
  - Stacking (rule 2)
- Any selection can be rewritten as an if statement, and any repetition can be rewritten as a while statement

Thank you!