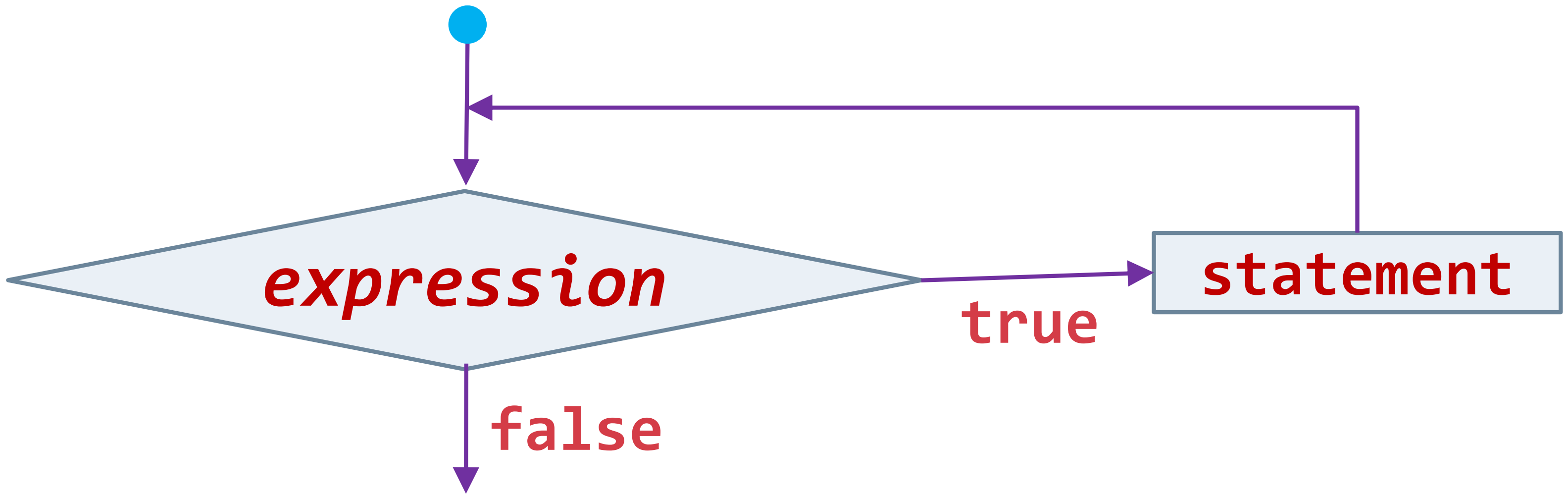
HIGH-LEVEL PROGRAMMING I

while Loop

by Prasanna Ghali

Iteration Structure

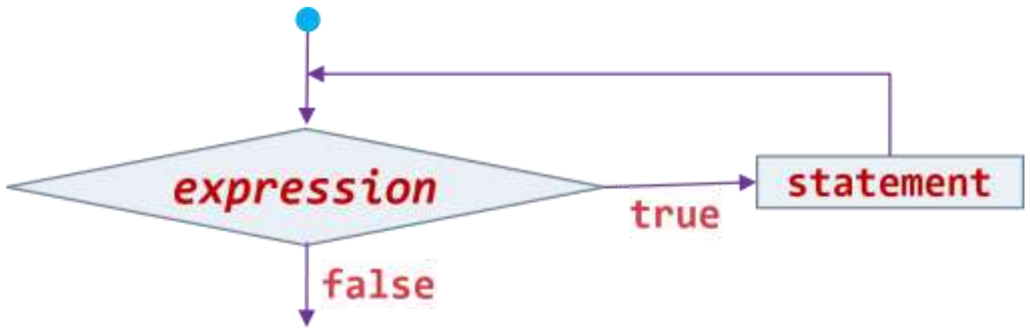
- Repeat actions *while* a condition remains true



```
while (expression)
    statement
```

statement or
block of **statement**s delimited by braces

while Loop (1/2)



```
int i = 0;

while (i < 20) {

  print("%d ", i);

  i += 5;
}
```

1 2 3

output

```
0 5 10 15
```

```
// initialize loop control variable(s)

// expression tests loop control variable
while (expression) {

   statement

   // update loop control variable

}
```

while Loop (2/2)

- What is printed to standard output and what is value of `i` after conclusion of loop?

```
int i = 0;
while (i <= 20) {
  print("%d ", i);
  i += 5;
}
```

Type 1: Counter-Controlled `while` Loops

- You know how many times certain things need to be done

```
// initialize N to specify how many
// times certain things need to be done

// initialize loop control variable
counter = 0;

// test loop control variable
while (counter < N) {
  // do the thing ...

  // update loop control variable
  counter += 1;
}
```

Type 1: Counter-Controlled `while` Loops: Computing Average

- You know how many times certain things need to be done

```
int counter = 0, sum = 0;
printf("Enter %d integers\n", N);
while (counter < N) {
  int temp;
  scanf("%d", &temp);
  sum += temp;
  counter += 1;
}

double average = (double)sum/N;
printf("sum: %d | average: %.2f\n", sum, average);
```

Type 1: Counter-Controlled `while` Loops: Checkerboard Pattern

- You know how many times certain things need to be done

```c
int main(void) {
  printf("Enter rows and cols: ");
  int rows, cols;
  printf("rows: %d | cols: %d\n", rows, cols);
  int r = 0;
  while (r < rows) {
    int c = 0;
    while (c < cols) {
      putc('*', stdout);
      c = c + 1;
    }
    putc('\n', stdout);
    r = r + 1;
  }
}
```

Type 2: Sentinel-Controlled `while` Loops

- You *don't know* how many times certain things need to be done

```c
int sentinel = -1;
printf("Enter integers ending with %d: ", sentinel);
int num;
scanf("%d", &num);

int sum = 0, count = 0;
while (num != sentinel) {
   sum += num;
   scanf("%d", &num);
   count += 1;
}
double average = (double)sum/count;
printf("sum: %d | average: %.2f\n", sum, average);
```

Type 3: Flag-Controlled while Loops

- Flag-controlled while loop uses boolean variable to control loop

```
// initialize loop control variable
bool found = false;

// test the loop control variable
while (!found) {
   ...
   // update loop control variable
   if (expression)
      found = true;
   ...
}
```

Type 3: Flag-Controlled while Loops – Number Guessing Game

```
srand(time(0)); // seed random number generator
int num = rand() % 100;
bool have_guessed = false;
while (!have_guessed) {
  printf("Enter a number between 1 and 100: ");
  int guess;
  scanf("%d", &guess);
  if (guess == num) {
    prinf("You guessed correct value: %d\n", guess);
    have_guessed = true;
  } else if (guess < num) {
    printf("Your guess is lower than number\n");
  } else {
    printf("Your guess is higher than number\n");
  }
}
```

Type 4: EOF Controlled `while` Loops (1/2)

- Algorithm to implement file copy by copying one character at a time from input to output file
    1) read character from input file
    2) while (character is not end-of-file indicator)
    3) write character read to output file
    4) read next character

Type 4: EOF Controlled while Loops (2/2)

- Flag-controlled while loop uses boolean variable to control loop

```c
#include <stdio.h>

int main(void) {
  int ch = getchar();
  while (ch != EOF) {
    putchar(ch);
    ch = getchar();
  }
  return 0;
}
```

```c
#include <stdio.h>

int main(void) {
  int ch;
  while ((ch = getchar()) != EOF) {
    putchar(ch);
  }
  return 0;
}
```

Infinite Loops

- If controlling expression never evaluates to `false`, you get an *infinite loop*
- Example 1:

```
int i = 1;
while (i != 10)
  i += 2;
```

- Example 2:

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

- Example 3:

```
while (1)
  printf("Infinite loop ...\n");
```