CPE101 Programming Languages I

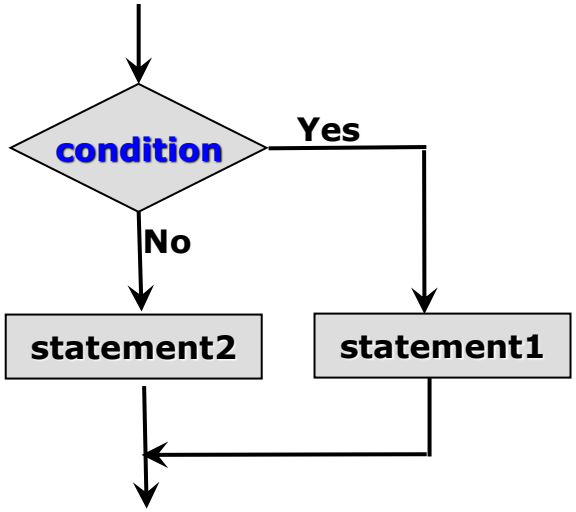
Week 9
Control and Loop Structures in C Language

Assoc. Prof. Dr. Caner ÖZCAN

Decision (Comparison) Commands if-else

- Commands used in the control of conditions.
- They direct the process flow depending on whether conditions are right (satisfy or not).

```
if (condition)
        statement1;
[else
        statement2;]
```



- If condition is true then the **true** (1) "Case 1" will run, otherwise "Case 2" will run.

if-else Examples

```
int finalGrade;

printf("Enter final grade: ");
scanf("%d", &finalGrade);
if (finalGrade >= 45)
   printf("Passed \n");
```

```
int finalGrade;

printf("Enter final grade: ");
scanf("%d", &finalGrade);
if (finalGrade >= 45)
   printf("Passed \n");
else
   printf("Failed!\n");
```

What should we do if we run more than one action?

block (associated states)

```
int finalGrade;
…
if( finalGrade >= 45)
{
    printf("Passed!\n");
    printf("Congrats!\n");
}
else
{
    printf("Failed!\n");
    printf("Work hard.\n");
} /* end-else */
```

Place of Fancy Brackets

- Place of fancy brackets is a style issue..
  - It is not a problem for the compiler.

```
int finalGrade;
…
if(finalGrade >= 45){
    printf("Passed!\n");
    printf("Congrats!\n");
} else {
    printf("Failed!\n");
    printf("Work Hard.\n");
} /* end-else */
```

Using the Logic Operator

- Logical operators can be used in the if statement.

```
/* If a is equal 4 OR a is equal 10 */
if (a == 4 || a == 10){
   ...
} else {
   ...
} /* end-else */
```

```
/* If x between 2 AND 20*/
if (x >= 2 && x <= 20){
   ...
} /* end-if */
```

```
/* If y is greater than 20 AND x is not equal 30 */
if (y > 20 && x != 30){
   ...
} /* end-if */
```

Logic Multistage if Statements

- Sometimes we want to test more than one condition, until one of them established.
  - For example we want to test "n" is equal to 0 , greater than 0 and lesser than 0.

```
if (n < 0)     printf("n < 0\n");
else {
  if (n == 0) printf("n == 0\n");
  else        printf("n > 0\n");
} /* end-else */
```

  - Rather than use second if statement in the else, we can use instead as follows. We called it logic multistage if.

```
if (n < 0)        printf("n < 0\n");
else if (n == 0) printf("n == 0\n");
else              printf("n > 0\n");
```

Multistage if Syntax

```
if (condition1)
       statement1;
[else if (condition2)
        statement2;
 else if (condition3)
        statement3;
 …
 else
        statementN;]
```

Multistage if Syntax

```
int finalGrade;
…
if (finalGrade >= 90)
       printf("Passed: Grade A \n");
else if (finalGrade >= 85)
       printf("Passed: Grade A- \n");
else if (finalGrade >= 80)
       printf("Passed: Grade B+ \n");
else if (finalGrade >= 75)
       printf("Passed: Grade B \n");
else if (finalGrade >= 70)
       printf("Passed: Grade B- \n");
else if (finalGrade >= 55)
       printf("Passed: Grade C \n");
else
       printf("Failed \n");
```

Condition Operator?

- Depending on the condition, related value or process result is transferred to the specified variable.

```
if  (condition) statement1;
else statement2;
```

- There is an operator that can write these statements shorter.
  - condition ? statement1 : statement2
- Examples:

```
min = (a < b) ? a : b;
```

takes minimum between a and b

```
a = (b >= 0) ? b : -b;
```

takes absolute value of b.

switch Statement

- Optional form structure.
  - In this case, we can use the following multistage if.

```
if (grade == 5)       printf("Perfect \n");
else if (grade == 4) printf("Good \n");
else if (grade == 3) printf("Pass\n");
else if (grade == 2) printf("Poor\n");
else if (grade == 1) printf("Fail\n");
else                  printf("Invalid Grade");
```

- As an alternative, instead of multistage if statements switch can be used in C.

switch Statement Example

- Rewriting the previous code using switch rather than multistage if.

```
switch(grade){
  case 5: printf("Perfect \n");
          break;
  case 4: printf("Good);
          break;
  case 3: printf("Pass\n");
          break;
  case 2: printf("Poor\n");
          break;
  case 1: printf("Fail\n");
          break;
  default: printf("Invalid Grade\n");
          break;
} /* end-switch */
```

switch Statement Syntax

- According to the condition in the statement, one option runs.

```
switch(condition) {
case value1:
        statement1;
        …
        [break;]
case value2:
         statement2;
        …
        [break;]
default:
         statementN;
        …
        [break;]
} /* end-switch */
```

switch Statement Example

```
char operator;
int result;
int value;
…
switch(operator) {
case '+':
          result += value;
        break;
case '-':
          result -= value;
        break;
case '*':
          result *= value;
        break;
case '/':
        if(value == 0) {
                printf("Error: dividing zero \n");
                printf("       operation is aborted  \n");
        }
        else
                  result /= value;
          break;
default:
        printf("unknown operation\n");
        break;
} /* end-switch */
```

break in switch

- "Break" throws us out of the switch block.
- If **break** is forgotten or not written, control continues to the next case.

```
switch(grade){
  case 5: printf("Perfect\n");
  case 4: printf("Good\n");
  case 3: printf("Pass\n");
  case 2: printf("Poor\n");
  case 1: printf("Fail\n");
  default: printf("Invalid grade\n");
} /* end-switch */
```

- If grade == 3, the following will write
  - Pass

    Poor

    Fail

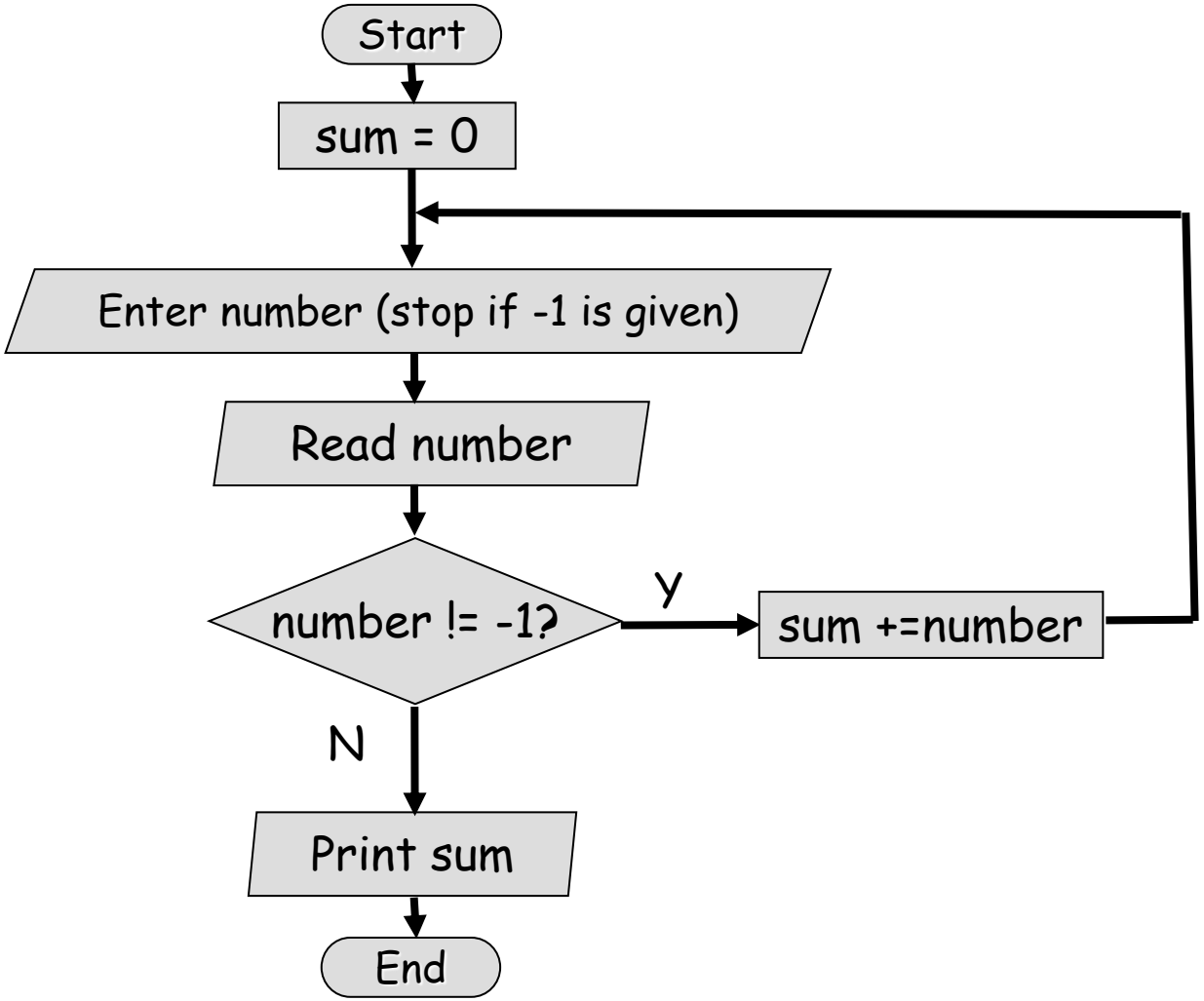
    Invalid grade

Loop Statements

- Commands that provides for making sequential or repetitive process.
- There are three loop structure in the C programming language.
  - while loops
  - do-while loops
  - for loops

- The following statement can be used to help quit in the loop.
  - break
  - continue

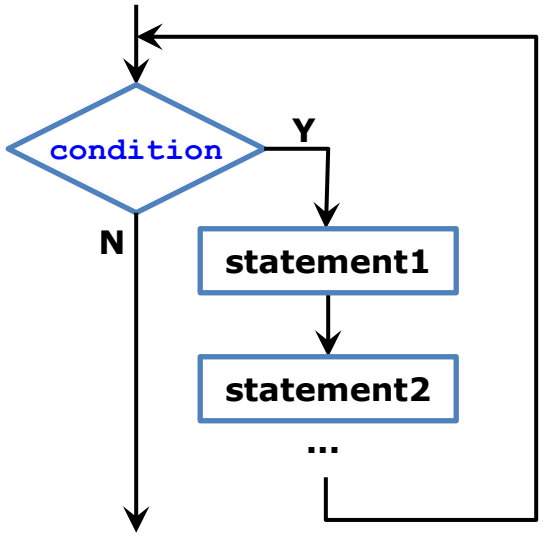
Calculation the sum of number series - Flowchart



while Statement

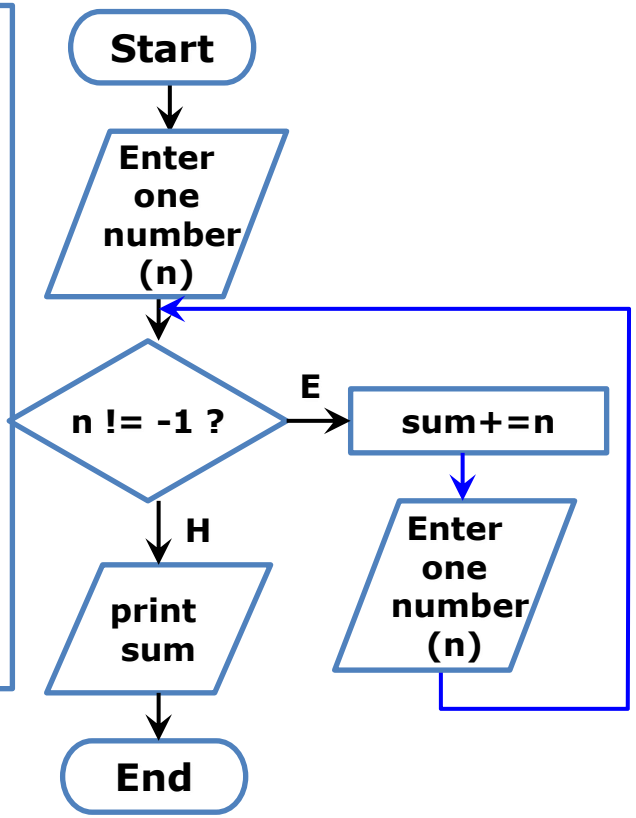
- Pre-conditional loop in the C language.
- Operations in the loop are performed as long as given condition is satisfied
- Used in situations if we do not know how many times loop needs to work.

```
while(condition)
{
        statement1;
        statement2;
        ...
}
```



Calculation of the total of the numbers entered from the keyboard

```
int sum = 0;
int n;

printf("Enter number(enter -1 stop): ");
scanf("%d", &n);

while (n != -1){
   sum += n;
   printf("Enter number(enter -1 stop): ");
   scanf("%d", &n");
}
printf("Sum =%d\n", sum);
```



while Example

```
int i = 0;

printf("How did you find the C programming?\n");
while(i < 10)
{
        printf("C Programming is very enjoyable!\n");
        i++;
}
```

- Loop 10 times (from 0 to 9)
- Print same message 10 times

while Example

```
int i = 20;

printf("How did you find the C programming?\n");
while(i < 10)
{
      printf("C Programming is very enjoyable!\n");
      i++;
} /* end-while */
```

- Loop 0 time (i = 20, not less than 10)
- No print within loop

while Example

- Problem: Calculate a^n

```
int i;
int num;
int pow;
int res=1;

printf("Enter Number: ");
scanf("%d", &num);
printf("Enter Power: ");
scanf("%d", &pow);

i = 1;
while (i<= pow)
  res *= num;
  i++;
} /*end-while*/
printf("num^pow = %d\n",res);
```

```
int i;
int num;
int pow;
int res=1;

printf("Enter Number: ");
scanf("%d", &num);
printf("Enter Power: ");
scanf("%d", &pow);

i = 1;
while (i++<= pow)
  res *= pow;

printf("num^pow = %d\n",res);
```

while Example

- Problem: Calculate 1+2+3+4+…+N

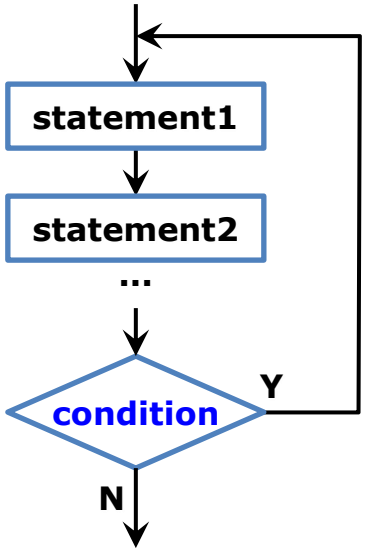
```
int i;
int n;
int sum = 0;

printf("Enter n:");
scanf("%d", &n);
i = 1;
while (i<= n){
  sum += i;
  i++;
} /* end-while */
printf("Sum: %d\n", sum);
```

```
int i;
int n;
int sum = 0;

printf("Enter n: ");
scanf("%d", &n);
i = 1;
while (i<= n)
  sum += i++;

printf("Sum: %d\n", sum);
```

do while Statement

- Final conditional loop.
- The loop is processed as long as the condition specified by "while" is satisfied (correct).
- In some cases we would like to decide whether or not to continue once the loop has been working. In these cases, the "do while" loop is used.
- The body of loop works at least once.

```
do
{
        statement1;
        statement2;
        ...
} while(condition);
```



do while Example

```
int i = 0;

printf("How did you find the C programming?\n");
do {
        printf("C Programming is very enjoyable!\n");
        i++;
} while(i < 10);
```

- Loop 10 times (from 0 to 9)
- Print same message 10 times

do while Example

```
int i = 20;

printf("How did you find the C programming?\n");
do {
        printf("C Programming is very enjoyable!\n");
        i++;
} while(i < 10);
```

- Loop 1 time (i = 20)
- Print same message 1 time

Differences between while and do while

while	do while
Controls to put the loop	Controls to exit from the loop.
Loop can run or not run	Loop works at least once

for Loop

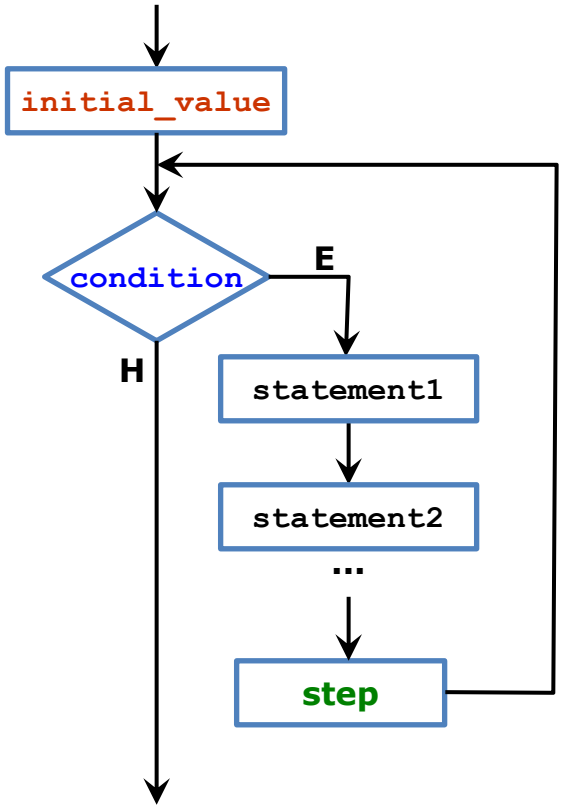
- It's a counter-loop command.
  - For example, we want to return N times.
- Used more often.
- Loop block operations are performed as long as the condition is true (correct).

```
for(initial_values; condition; step)
{
        ...
        transactions
        ...
}
```

Flowchart of for loop and while pair

```
for(initial_value;condition;step)
{
        statement1;
        statement2;
        ...
}
```

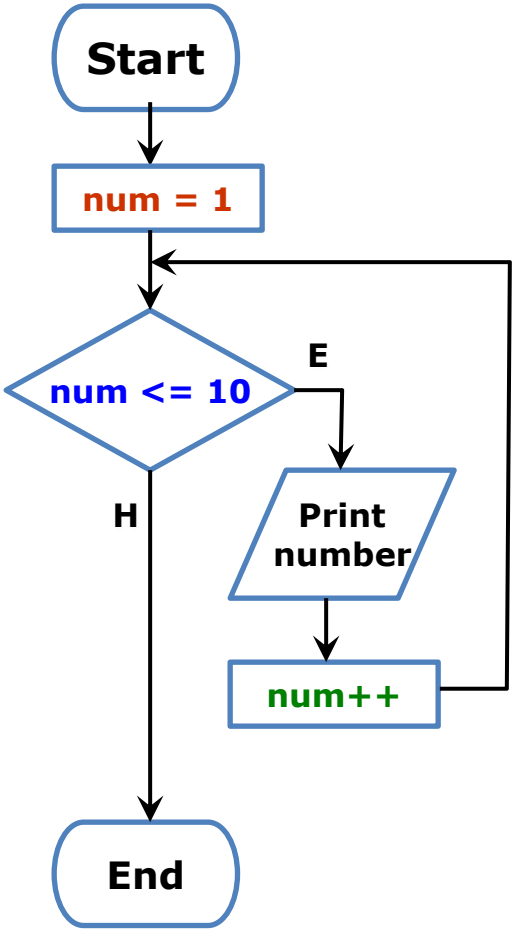
```
initial_value;
while(condition)
{
        statement1;
        statement2;
        ...
        step;
}
```



for Example

- Problem: print numbers from 1 to 10

```
int num;
for(num = 1; num <= 10; num++)
{
     printf("%d ",num);
}
printf("\n");
```



output:

```
1 2 3 4 5 6 7 8 9 10
```

Use of For Statement

- For statement is usually the best choice to increase or decrease a variable.

```
for (i=0; i<N; i++) …
```

- Count from 0 to N-1.

```
for (i=1; i<=N; i++) …
```

- Count from 1 to N.

```
for (i=N-1; i>=0; i--) …
```

- Count from N-1 to 0.

```
for (i=N; i>=1; i--) …
```

- Count from N to 1.

for Example

- Problem: Calculate 1 + 2 + 3 + 4 + ... + N operations

```
top = 0;
for(i=1; i<=N; i++){
   top += i;
}
```

- The first value (i = 1), the control (i <= N) and replace (i ++) expressions are optional and can not be typed.

```
sum=0;
i=1;
for(; i<=N; i++){
    sum += i;
}
```

```
sum=0;
for(i=1; i<=N;){
    sum += i++;
}
```

```
sum=0;
i=1;
for(; i<=N;){
    sum += i++;
}
```

Inner Loops

- We can use loops within one
  - Most programs require this.
- Example: print the multiplication table

```
int i, j;

for (i=1; i <= 10; i++){
  printf("i: %d: ",i);
  for (j=1; j <= 10; j++){
    printf("%5d", i*j);
  } /* end-for-inner*/
  printf("\n");
} /* end-for-outer*/
```

break & continue

- In the case of switch, we saw the break code throw us out of the switch statement.
- Similarly, if you are used in a break loop, it will throw us out of the current loop.

```
int sum = 0;
int n;

while (1){  /* endless loop*/
  printf("enter a number (to stop enter -1): ");
  scanf("%d", &n");

  if (n == -1) break;  /* go outside of the loop */
  sum += n;
} /* end-while */

printf("sum=%d\n", sum);
```

break

- In some cases it is used to exit the loop midway through the turn.

```
while (1){
  printf("enter a number and 0 for end: ");
  scanf("%d", &n);
  if (n == 0) break;
  printf("n=%d, n*n*n*=%d\n", n, n*n*n);
} /* end-while */
```

continue

- Take control at the end of the loop.
- Do not forget, we're still in loop.
- "Continue" simply skips the next part of the body that is used in the body of the return, and take the control at the end of the body.

```
int i;
int n = 0;
int sum = 0;

while (n<10){
  scanf("%d", &i);
  if (i == 0) continue; /*move to (B) */
  n++; sum += i;

  /*(B)*/
} /* end-while */
```

Endless Loops

- Unlimited number of repeating loops.

```
while (1){
   …
}
```

```
do{
  …
} while (1);
```

```
for (;;){
   …
}
```

- How can we get out of this cycle?
  - Simply by putting a "break" somewhere around the loop.

```
while (1){
  printf("Enter a number (to stop enter 0): ");
  scanf("%d", &n);
  if (n == 0) break;
  printf("n=%d, n*n*n*=%d\n", n, n*n*n);
} /* end-while */
```

Comma Operator

- From time to time we want to combine several expressions and write them as a single expression.
  - In such cases, we use the comma operator.

```
statement1, statement 2, …, statement N;
```

```
i=1, j=2, k=i+j;
```

Equal to

```
((i=1), (j=2), (k=i+j));
```

  - Evaluation is done from left to right
  - Expression result = k=i+j; so 3

References

- Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- J. G. Brookshear, “Computer Science: An Overview 10th Ed.”, Addison Wisley, 2009.
- Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- Paul J. Deitel, “C How to Program”, Harvey Deitel.
- Bayram AKGÜL, C Programlama Ders notları