Flowchart

1. Do you have a problem in your life?
   - No: Then don’t worry
   - Yes: Can you do something about it?
     - No: Then don’t worry
     - Yes: Then do something about it
Example - Review

An airplane is accelerating smoothly for 15 minutes at a speed of 480 km / min. Then goes steady for 20 minutes. It then slows down smoothly for 15 minutes and the speed is zero. Plot the flow diagram of the algorithm giving the speed at any time.
Başla

t

t<15
E
V=32*t

>15
H

t>35
E

>35
H

V=480

V=32*t

t=50-t

V

Bitir
Loop Structures

► A loop statement allows to specify that an action is to be repeated while some condition remains true.
► For example, as long as the products have not been on a shopping list in hand, you can continue shopping.
► The condition "Remaining items on the list?" may be true or false.
► If it is true "Purchase next item and cross it off from my list" is performed. This action will be performed repeatedly while the condition remains true.
In the following pseudo code, value of result starts by 2 and multiplied by 2 in each step while it is less than 1000.

Pseudo code:

1. Start
2. result = 2
3. while result < 1000
   result = result * 2
4. End
Here the action is carried out as long as the condition is provided in the decision symbol and return to the structure of control again.

When condition is not provided, the loop is exited and control passes to the next statement in the program.

When product becomes 1024 the condition in the while statement product <= 1000 becomes false.

This terminates the repetition and the final value of product is 1024.
Loop Structures

There are two types of loop structures:

– Counter-controlled loop
– Sentinel-controlled loop
Counter-controlled Loop Structure

- **Counter-controlled** loop is sometimes called definite repetition because exactly we know how many times the loop will be executed.

- In counter-controlled loop, a control variable is used to count the number of repetitions.

- The control variable is incremented (usually by 1) in each step of the loop.

- When the value of the control variable indicates that the required number of repetitions has been performed, the loop terminates and the computer continues to execute commands after the loop statement.
Counter-controlled Loop Structure

Counter-controlled loop requires

– The name of a control variable (or loop counter)
– The initial value of the control variable
– Increasing (or decreasing) of the control variable at each step of the loop
– A condition statement that tests the final value of the control variable
Consider the following problem:

– A class of ten students took a quiz. The grades (integers in the range 0-100) for this quiz are available. Determine the class average on the quiz.

The class average is equal to the sum of the grades divided by the number of students.

We should find the sum of student grades and divide it by 10 to calculate the average.

In each step one grade of a student is entered and grade entering operation will be repeated 10 times due to the number of students is certain.
Counter-controlled Loop Structure Example 2

Because a certain number of repetitions counter-controlled loop structure can be used in this problem.

This technique uses a variable called a counter to specify the number of times a set of statement should execute.

To keep track of the number of repetitions in such loops, one counter value is controlled with increasing by 1 at each step.

A total is find by adding a collection of grades entered in each step.
Counter-controlled Loop Structure Example 2

- **Total** and **counter** variables should normally be **initialized** before being used in a program.
- Otherwise the sum would include the previous value stored in the memory location total variable.
- **Counter** variables are normally initialized to **zero** or **one** depending on their use.
We approach the class average program with a technique called top-down stepwise refinement.

This is a technique that is essential for the development of well-structured programs.

- Begin with a pseudo code representation of the top:
  
  Calculate the average quiz grade of class

- Divide top into smaller tasks and list them in order:
  
  Initialize variables
  
  Enter the exam notes, sum and count
  
  Calculate and print the class average

Problem Solution with Top-Down Stepwise Refinement Example 2
Problem Solution with Top-Down Stepwise Refinement Example 2

- Let's divide initial value assignment step to the smaller steps:
  - Initialize total to zero
  - Initialize counter to zero

- Let’s refine enter the exam notes, sum and count step:
  - Repeat while grade counter is less than ten
    • Enter grade
    • Add entered grade to the sum
    • Increase counter variable by 1
Problem Solution with Top-Down Stepwise Refinement

Example 2

Let’s refine *calculate and print the class average* step:

– *Calculate the average by dividing the total variable to the counter variable*

– *Print the average*
Problem Solution with Top-Down Stepwise Refinement

Example 2

– Initialize total to zero
– Initialize counter to zero

– Repeat while grade counter is less than ten
  • Enter grade
  • Add entered grade to the sum
  • Increase counter variable by 1

– Calculate the average by dividing the total variable to the counter variable
– Print the average
1. Start
2. total = 0, counter = 0
3. while counter < 10
   enter student grade, grade
   total = total + grade
   counter = counter + 1
4. average = total / counter
5. Print, average
6. End
 Sentinel-controlled Loop Structure

- **Sentinel-controlled** repetition is sometimes called **indefinite repetition** because it’s not known in advance how many times the loop will be executed.

- **Sentinel value** (also called a **signal value**, or a **flag value**) indicates “end of data entry.”

- The sentinel is entered after all regular data items have been supplied to the program.

- The sentinel value must be chosen so that it cannot be confused with an acceptable input value.

- Sentinel values are used to control repetition when:
  - The precise number of repetitions are not known in advance,
  - If the data entry from user is performed at every step of the loop.
Sentinel-controlled Loop Structure Example 3

► Consider the following problem:
  —Develop a class averaging program that will process an arbitrary number of grades each time the program is run.

► In this example, the program must process an arbitrary number of grades.

► How can the program determine when to stop the input of grades? How will it know when to calculate and print the class average?
Clearly, we can not use counter-controlled repetition structure as we don't know the number of students. Instead of counter-controlled repetition, we have to use sentinel-controlled repetition structure.

The sentinel value must be chosen so that it cannot be confused with an acceptable input value.

Since grades on a quiz are normally nonnegative integers, –1 is an acceptable sentinel value for this problem.
Problem Solution with Top-Down Stepwise Refinement

Example 3

Top-Down Stepwise Refinement

– Begin with a pseudo code representation of the top:

Determine the class average for the quiz

– Divide top into smaller tasks and list them in order:

Initialize variables
Input, sum, and count the quiz grades
Calculate and print the class average
Problem Solution with Top-Down Stepwise Refinement

Example 3

► Refine the initialization step:
  – Initialize total to zero
  – Initialize counter to zero

► Refine input, sum and count the quiz grades step:
  – Input the first grade (possibly the sentinel)
  – Repeat unless entered -1 from the keyboard
    • Add this grade into the total variable
    • Increase a counter variable (counter).
    • Input the next grade (possibly the sentinel)
Refine **Calculate and print the class average** step

- If the counter is not equal to zero
  - Calculate the average by dividing the total variable to the variable count
  - Print the average
- else
  
  Print “No grades were entered”
Problem Solution with Top-Down Stepwise Refinement

Example 3

– Initialize total to zero
– Initialize counter to zero

– Input the first grade (possibly the sentinel)
– Repeat unless entered -1 from the keyboard
  • Add this grade into the total variable
  • Increase a counter variable (counter).
  • Input the next grade (possibly the sentinel)

– If the counter is not equal to zero
  • Calculate the average by dividing the total variable to the variable count
  • Print the average
– else
  Print “No grades were entered”
1. Start
2. total = 0, counter = 0
3. Enter student grade, grade
4. while grade != -1
   total = total + grade
   counter = counter + 1
   Enter student grade, grade
5. If counter==0 then
   average = total / counter
   Print, average
else
   Print, "Grade is not entered"
6. End
Develop an algorithm that prints all divisors of the given number from keyboard.

1. Start
2. Read N from keyboard
3. X = 1
4. while X <= N
   if N % X == 0
      print X
     Add one to X
5. End
We have 10 numbers entered from the keyboard and we will calculate the number of the odd and even ones individually.

1. Start
2. evenCounter = 0
3. oddCounter = 0
4. counter = 1
5. while counter \leq 10 repeat
   Enter number from keyboard, number
   if number \% 2 == 0 then
     evenCounter ++
   else
     oddCounter ++
   counter++
6. Print evenCounter
7. Print oddCounter
8. End
Create a flow chart of a program that finds number entered from the keyboard is perfect or not.

- Perfect number: Except itself, the sum of the divisor equals the number itself

- For example: $6 \rightarrow 1+2+3 = 6$
1. Start
2. Enter number from keyboard, number
3. total = 0
4. counter = 1
5. while counter < number repeat
   if number % counter == 0
   sum += counter
   counter++
6. if sum == number
   print "Perfect number"
7. else
   printf "Not perfect number"
8. End
References


► Paul J. Deitel, “C How to Program”, Harvey Deitel.

► Bayram AKGÜL, C Programlama Ders notları