Problem Solving using Loops

Rupesh Nasre.

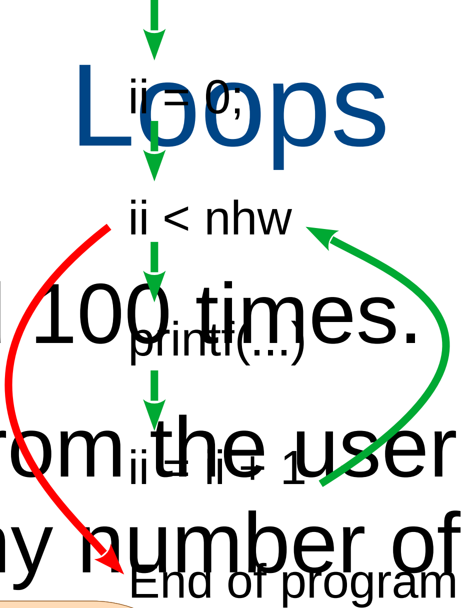
IIT Madras

November 2022

Assignments and Conditionals are not enough!

- We need to repeat a processing.
- Finding the largest of a bunch of numbers
- Finding if a large number of roll numbers are from our department
- Will every student pass this course?
- Three types of loops in C
  - for
  - while // for and while are equivalent
  - do-while

Loops



- Print Hello World 100 times.
- Take a number from the user and print Hello World those many number of times.

```
int main() {
        int nhw;
        scanf("%d", &nhw);
        int ii;
        ii = 0;
NextIteration:
        if (ii < nhw) {
                printf("Hello World\n");
                ii = ii + 1;
                goto NextIteration;
        }
}
```

```
int main() {
        int nhw;
        scanf("%d", &nhw);
        int ii;
        for (ii = 0; ii < nhw; ii = ii + 1)
                printf("Hello World\n");
}
```

No semicolon

Can also be written as

for (ii = 1; ii <= nhw; ii = ii + 1)

Problem: Check Pass %

- For each student, find if total marks >= 40.

```
#define NSTUDS 87

pass = 0;
for (int stud = 0; stud < NSTUDS; ++stud) {
        scanf("%d", &marks);
        if (marks >= 40) {
                printf("Pass\n");
                pass++;
        }
}
printf("%f%%\n", (float)pass / NSTUDS);
```

Increment by 1.
Same as stud = stud + 1

break brings control out of the immediately enclosing switch or loop.

```
int stud = 0;
for (; stud < NSTUDS; stud++) {…} ✓
```

```
int stud = 0;                          ✗
for (;; stud++)
        if (stud < NSTUDS) {…}
        else break; ✓
```

These two are infinite loops.

```
int stud = 0;
for (;;)
        if (stud < NSTUDS) {
                …                      ✗
                ++stud;
        } else break; ✓
```

Problem: Print divisors

One may think of adding a check for negatives and zero, but the code may work as it is.

num iterations

```
scanf("%d", &num);
for (int div = 1; div <= num; ++div) {
	if (num % div == 0) {
		printf("%d\n", div);
	}
}
```

But these two codes are not equivalent. Why?

```
30
1
2
3
5
6
10
15
30
```

```
canf("%d", &num);
or (int div = num; div > 0; --div) {
	if (num % div == 0) {
		printf("%d\n", div);
	}
```

```
--div
div--
div = div – 1
div -= 1
```

Print the divisors in the descending order.
How about doing it for all the numbers 1 to 100?

Problem: Print divisors of 1..100

```
for (int num = 1; num <= 100; ++num) {
    printf("Divisors of %d are\n", num);
    for (int div = 1; div <= num; ++div) {
        if (num % div == 0) {
            printf("%d\n", div);
        }
    }
}
```

Outer loop
Inner loop
Nested loop

Number of div iterations increases with num.

I am interested only in odd numbers.

```
for (int num = 1; num < 100; num += 2)
```

I am interested only in odd divisors.

```
for (int div = 1; div <= num; ++div)
    if (num % div == 0 && div % 2 == 1)
```

OR

```
for (int div = 1; div <= num; div += 2)
    if (num % div == 0)
```

What happens in this case?

```
for (int div = 1;
    div <= num && num % div == 0;
    num += 2)
```

Problem: Print calendar

We already know how to find month days (switch).

Let's print those days with %4d for each week.

We will assume that the month starts on Monday, and February has 28 days.

```
for (int day = 1; day <= ndays; ++day) {
        printf("%4d", day);
        if (day % 7 == 0) printf("\n");
}
```

```
   1   2   3   4   5   6   7
   8   9  10  11  12  13  14
  15  16  17  18  19  20  21
  22  23  24  25  26  27  28
  29  30  31
```

Extend it to print it for the whole year (nested loop).
Can you make the next month start on the appropriate day (assume January 1 as Monday)?

Problem: Print patterns

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

Problem: Check Pass %

- For each student, find if total marks >= 40.

```
pass = 0;
nstuds = 0;
scanf("%d", &marks);
while (marks >= 0) {
        if (marks >= 40) {
                printf("Pass\n");
                pass++;
        }
        ++nstuds;
        scanf("%d", &marks);
}
printf("%f%%\n", (float)pass / nstuds);
```

At least one variable in the loop condition must change in the loop (sans break / goto).

Do this in batches, while the user is free to enter the marks.

```
pass = 0;
nstuds = 0;
while (marks >= 0) {
        scanf("%d", &marks);
        if (marks >= 40) {
                printf("Pass\n");
                pass++;
        }
        ++nstuds;
}
printf("%f%%\n", (float)pass / nstuds);
```

Negative marks are also getting processed.

One more than the number of students.

Two options:
1. Ask user every time if she is free.
2. Derive it based on marks entered.

Problem: 3n + 1

Also called as
Collatz Conjecture.

If the input positive number n is 1 then stop; else if it is even, halve it; else increase it to 3n+1. Repeat.

```
scanf("%d", &n);
while (n != 1) {
        if (n % 2 == 1)       // odd
                n = 3 * n + 1;
        else                  // even
                n /= 2;
        printf("%d\n", n);
}
printf("How do I always get printed?\n");
```

```
20
10
5
16
8
4
2
1
```

```
7
22
11
34
17
52
26
13
40
20
10
...
```

```
9
28
14
7
...
```

Problem: Mini-calculator. Repeated.

Given an expression num#num, print its value. # is +, -, *, /.

Read %d%c%d, stop when the operator is =.

```
scanf(
swit
");
}
```

```
scanf("%d%c%d", &x, &op, &y);
for (; op != '=';) {
    switch (op) {
        case '+': printf("%d\n", x + y); break;
        case '-': printf("%d\n", x - y); break;
        case '*': printf("%d\n", x * y); break;
        case '/': if (y != 0) printf("%d\n", x / y);
                  else printf("Division by zero error\n");
                  break;
        default: printf("Invalid operator %c\n", op);
    }
    scanf("%d%c%d", &x, &op, &y);
}
```

```
while (op != '=') {
    scanf("%d%c%d", &x, &op, &y);
    switch (op) {
        case '+': printf("%d\n", x + y); break;
        case '-': printf("%d\n", x - y); break;
        case '*': printf("%d\n", x * y); break;
        case '/': if (y != 0) printf("%d\n", x / y);
                  else printf("Division by zero error\n");
                  break;
        default: printf("Invalid operator %c\n", op);
    }
}
```

But didn't we say that while and for are equivalent?

```
for (ii = 0; ii < N; ++ii) {
    // statements
}
```

```
ii = 0;
while (ii < N) {
    // statements
    ++ii;
}
```

Problem: Print digits

Future Connect:
Printing the digits in the correct order would need array, stack, or recursion.

Print digits of a given positive (>0) integer.

```
scanf("%d", &n);
while (n > 0) {
        printf("%d\n", n % 10);
        n = n / 10;
}
```

```
#define BASE 10
scanf("%d", &n);
while (n > 0) {
        printf("%d\n", n % BASE);
        n = n / BASE;
}
```

```
#define BASE 16
scanf("%d", &n);
while (n > 0) {
  rem = n % BASE;
  if (rem < 10) printf("%d\n", rem);
  else printf("%c\n", 'A' + rem - 10);
  n = n / BASE;
}
```

Visualize this code

How about printing 10 as A, 11 as B, ...?

Extend the code for 0.
Extend the code for negative integers.

Problem: Numerics

Future Connect:
These could be naturally modeled using recursion.

Print sums: for n, print Σ1, Σ2, Σ3, ..., Σn

Factorial n! = n*(n-1)*(n-2)*...*2*1

Fibonacci $F_n = F_{n-1} + F_{n-2}$, $F_1 = 0$, $F_2 = 1$

```
sum = 0;
for (int ii = 1; ii <= n; ++ii) {
        sum += ii;
        printf("%d ", sum);
}
```

```
fact = pow2 = 1;
for (int ii = 1; ii <= n; ++ii) {
        fact *= ii;
        pow2 *= 2;
        printf("%ld %d ", fact, pow2);
}
```

```
fibprev = 1, fibprevprev = 0;
for (int ii = 1; ii <= n; ++ii) {
        fib = fibprev + fibprevprev;
        fibprevprev = fibprev;
        fibprev = fib;
        printf("%d ", fib);
}
```

Problem: Denominations

fixed number of user-defined number of

You are given ~~infinite~~ notes of ~~three~~ denominations as input. You need to find out if you have exact money to pay for an item purchased.

```
scanf("%d", &price);
scanf("%d%d%d", &deno1, &deno2, &deno3);
for (int d1 = 0; d1 <= price / deno1; ++d1)
    for (int d2 = 0; d2 <= price / deno2; ++d2)
        for (int d3 = 0; d3 <= price / deno3; ++d3)
            if (d1 * deno1 + d2 * deno2 +
                d3 * deno3 == price)
                printf("%d %d %d", d1, d2, d3);
```

55	550	18
10	10	7
20	20	11
100	100	13
No	1 2 5	1 1 0

d1 < min(price/deno1, maxdeno1)

How to write user-defined number of iterations?

Problem: Print digits

Print digits of a given positive (>0) integer.

```
scanf("%d", &n);
while (n > 0) {
        printf("%d\n", n % 10);
        n = n / 10;
}
```

```
scanf("%d", &n);
do {
        printf("%d\n", n % 10);
        n = n / 10;
} while (n > 0);
```

$$\begin{tikzcd} {} \arrow[d] \\ \text{scanf(...)} \arrow[d] \\ \text{printf(...)} \arrow[d] \\ \text{n = n / 10} \arrow[d] \\ \text{n > 0} \arrow[d] \arrow[uu, bend right] \\ \text{End of program} \end{tikzcd}$$

Since the input is always >0,
it is guaranteed that the loop
iterates at least once.

```
        for (ii = 0; ii < N; ++ii) {
                // statements
         }
```

$\Longleftrightarrow$

```
ii = 0;
if (ii < N)
        do {
                // statements
                ++ii;
        } while (ii < N);
```

Problem: Find the winner.

Given a log of who won each match, find the winner.

```
d = n = 0;
do {
      scanf("%c", &player);
      if (player == ‘D‘) d++;
      else if (player == ‘N‘) n++;
      else if (player == ‘X‘) break;
} while (1);
if (d > n) printf("Djokovic wins\n");
else if (d < n) printf("Nadal wins\n");
else printf("It‘s a draw\n");
```

```
d = n = 0;
for (;;) {
      scanf("%c", &player);
      if (player == ‘D‘) d++;
      else if (player == ‘N‘) n++;
      else if (player == ‘X‘) break;
}
if (d > n) printf("Djokovic wins\n");
else if (d < n) printf("Nadal wins\n");
else printf("It‘s a draw\n");
```

Write it using a do-while.

Write it using a for loop.

As a good programming practice, break should be used sparingly.

Problem: Find the GCD.

Given positive integers x and y, find their GCD.

Choosing the right algorithm is crucial.

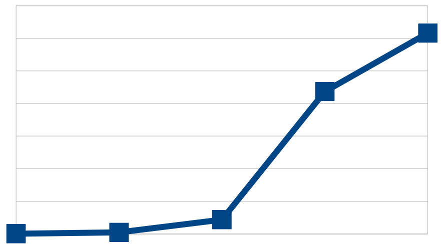
// find prime factorization of x and y
// multiply the common factors

Issue: Need to store factors (need array)
Another issue: How to find factorization?

For x = 432432331, y = 432432437,
time taken = 583 ms

```
if (x > y) min = y; else min = x;

for (int ii = min; ii > 0; --ii)
        if (x % ii == 0 && y % ii == 0) {
                printf("GCD = %d\n", ii);
                break;
        }
```



It is guaranteed due to the properties of numbers that the break gets executed for positive integers.

Example borrowed from Meghana Nasre.

Problem: Count frequencies

- Given a line, count the number of capital, small-case, digits, whitespaces.

```
int capital = 0, smallcase = 0, digit = 0, whitespace = 0, nread = 0;
char c;

while ((c = getchar()) != '\n') {
        if (c >= 'A' && c <= 'Z') capital++;
        else if (c >= 'a' && c <= 'z') smallcase++;
        else if (c >= '0' && c <= '9') digit++;
        else if (c == ' ' || c == '\t') continue;
        printf("Number of useful chars read = %d\n", ++nread);
}
printf("The line contains %d capital letters, %d small-case letters, "
        "%d digits\n", capital, smallcase, digit);
```

continue goes back
to the condition,
ignoring the rest of the loop.

continue

```
for (ii = 0; ii < 10; ++ii) {
        printf("pre  ii = %d\n", ii);
        if (ii % 2) continue;
        printf("post ii = %d\n", ii);
}
```

```
pre  ii = 0
post ii = 0
pre  ii = 1
pre  ii = 2
post ii = 2
pre  ii = 3
pre  ii = 4
post ii = 4
pre  ii = 5
pre  ii = 6
post ii = 6
pre  ii = 7
pre  ii = 8
post ii = 8
pre  ii = 9
```

```
ii = 0;
while (ii < 10) {
        printf("pre  ii = %d\n", ii);
        if (ii % 2) continue;
        printf("post ii = %d\n", ii);
        ++ii;
}
```

```
pre  ii = 0
post ii = 0
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
pre  ii = 1
…
```

```
do {
        printf("pre  c=%c\n", c);
        if (c == ' ') continue;
        printf("post c=%c\n", c);
} while ((c = getchar()) != '\n');
```

```
pre   c=
post c=
He llo
pre   c=H
post c=H
pre   c=e
post c=e
pre   c= 
pre   c=l
post c=l
pre   c=l
post c=l
pre   c=o
post c=o
```

Problem: Secret Message

- Given a text, encrypt it (and decrypt).

```
if (input == 'e') inc = 1;
else if (input == 'd') inc = -1;
...

char c;
while ((c = getchar()) != '\n') {
        c += inc;
        putchar(c);

 }
```

```
if (input == 'e') {
        char c = getchar();
        while (c != '\n') {
                c += 1;         // -= for decrypt
                putchar(c);
                c = getchar();
         }
} else if (input == 'd') {
        ...
}
```

Can we avoid this code duplication?

```
Hello World!
Ifmmp!Xpsme"
```

Need to have separate binaries for encrypt and decrypt. Alternatively, we can ask the user what she wants to do.

Multiple ways:
1. Use #define. Pass + or - as a parameter.
2. Use if-else or ?: inside the loop.
3. Decide +1 / -1 a priori, and use that in loop.

Problem: Menu

```
#define mycase(op, opchar)    printf("Enter the numbers: "); \
                              scanf("%d%d", &x, &y); \
                              printf("%d %c %d = %d\n\n", x, opchar, y, x op y); \
                              break;
int main() {
    enum {Exit, Add, Subtract, Multiply, Divide};
    int operation, x, y;
    do {
    // display menu
        printf("1: Add\n2: Subtract\n3: Multiply\n4: Divide\n0: Exit\n\nYour choice? ");
    // get input
        scanf("%d", &operation);
    // perform operation
        switch (operation) {
        case Add: mycase(+, '+');
        case Subtract: mycase(-, '-');
        case Multiply: mycase(*, '*');
        case Divide: mycase(/, '/');
        case Exit: break;
        default: printf("Invalid operation\n");
        }
    // repeat
    } while (operation != Exit);
}
```

	Week	Problems	Tools
✓	0	Solve equations, find weighted sum.	Data types, expressions, assignments
✓	1	Find max, convert marks to grade.	Conditionals, logical expressions
✓	2	Find weighted sum for all students.	Loops
	3	Encrypt and decrypt a secret message.	Character arrays
	4	Our first game: Tic-tac-toe	2D arrays
	5	Making game modular, reuse.	Functions
	6	Find Hemachandra/Fibonacci numbers.	Recursion
	7	Encrypt and decrypt many messages.	Dynamic memory, pointers
	8	Maintain student records.	Aggregate data types
	9	Search and sort student records.	Searching and sorting algorithms
	A	Reduce memory wastage.	Linked lists
	B	Implement token system in banks.	Queues
	C	IRCTC-like ticket booking system	File handling
	D	Putting it all together	All the above