Short Course on Programming in C/C++

Organized by Onur Pekcan

Contributor Selim Temizer

Instructor Hasan Yılmaz

MİTENG
MITAS ENGINEERING INC.
Eski Güvercinlik Yolu No : 113 Gazi 06560 Ankara TURKIYE
P: + 90 312 296 2000 :: f: + 90 312 296 2999
www.miteng.com

Applied Innovative Interdisciplinary (AI2) Research Lab
www.ai2lab.org

Mobility Research Lab
mobility.ceng.metu.edu.tr

Week 1 - Lecture 1

Today

We will cover;

- Overview of Programming Languages
- Introduction to C, “Hello World”
- Data Types and Expressions
- Control Flow

Overview of Programming Languages

- Functional Languages
  - Data environment is restricted
  - Functions recieve their parameters and return their result
  - No data regions are created
  - No Assignment
  - In fact, variables are functions
  - Higher-Order Functions
    - f0g(x) -> f(g(x))
  - Recursion
    - f(1) = 1
    - f(x) = f(x-1).(2x+1)
  - Problems are solved using only functions
  - Python, Haskell etc.

Overview of Programming Languages

- Imperative Languages
  - Problem is solved by writing down a sequence of action units which are called statements.
  - Each statement performs either a change on the data environment of the program or changes the flow of execution.
  - Imperative programs are easy to translate to machine code
  - If statement1 is followed by statement2, in the machine code translations of these statements machine_code1 will also be followed by machine_code2
  - C, C++, Java, Php, Python etc.

Overview of Programming Languages

- Logic Programming
  - In this paradigm, the programmer states the relations among the data as facts or rules(also referred as relations)
  - For example, facts can be information about who is whose mother and the rule can be a logical rule
  - Below is such a logical program in Prolog a well-known logical programming language.

```
mother(matilda, ruth).
mother(trudi, peggy).
mother(eve, matilda).
mother(eve, trudi).
grandmother(X,Y) :- mother(X,Z), mother(Z,Y).
?- grandmother(G,T).
                    G = eve, T = ruth
                    G = eve, T = peggy
?- grandmother(eve, matilda).
                    False.
?- grandmother(eve, X).
                    X = ruth
                    X = peggy
```

  - Prolog

Overview of Programming Languages

- Object-Oriented Programming
  - The most common paradigm in commercial circles
  - Data and Action of Data are not seperated
  - An Object has some internal data and functions, so called methods.
  - Possible to create as many instances of an object
  - We will cover this paradigm later in C++

  - C++, Java, Php, Python etc.

Introduction to C, “Hello World”

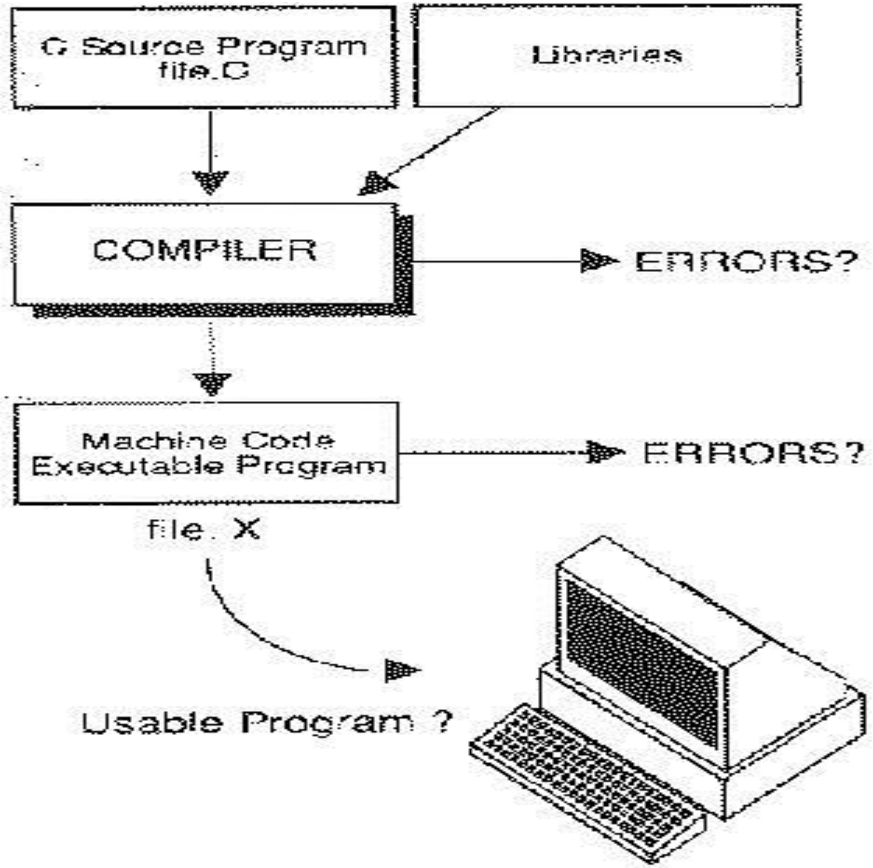


Figure 1.1. The stages of compilation.

Introduction to C, “Hello World”

- Printing "Hello World!"

```
#include<stdio.h>

int main()
{

    printf("Hello World!\n");

return 0;
}
```

Data Types, Expressions

- Data types
- Type Conversion
- Basic I/O
- Arithmetic and Logical Expressions
- Assignment
- Statements

Data Types, Expressions

- **Data Types**
  - Integer
  - String
  - Char
  - Float

Data Types, Expressions

- **Integer constants**
  - short
  - int
  - unsigned int
  - long int
  - unsigned long int

Data Types, Expressions

- **String Constants**
  - “deneme bir iki”
  - “deneme bir iki”\
  - “uc dort bes”
  - “deneme \” bir iki”

Data Types, Expressions

- **Character Constants**
  - ‘a’, ‘1’, ‘%’, …
  - ‘\’’
  - ‘c’ vs “c”

Data Types, Expressions

- **Floating Points**
  - float
  - double
  - long double

Data Types, Expressions

Type	Bytes	Range
short int	2	-32,768 -> +32,767
unsigned short int	2	0 -> +65,535
unsigned int	4	0 -> +4,294,967,295
int	4	-2,147,483,648 -> +2,147,483,647
long int	4	-2,147,483,648 -> +2,147,483,647
signed char	1	-128 -> +127
unsigned char	1	0 -> +255
float	4	
double	8	
long double	12	

Data Types, Expressions

Declaration:

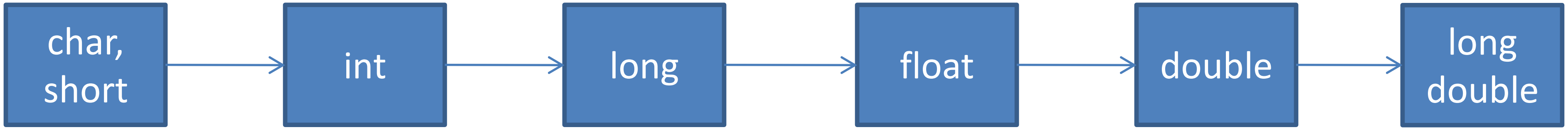
```
int a;
char c;
```

Initialization:

```
float x = 0.34;
int y = 2345;
```

Type Conversion

- Automatic Type Conversion Rules



char, short → int → long → float → double → long double

Example: *c -> int, f -> float*

- c/f *result -> float*

** Advice: Avoid automatic type conversion!*

- Explicit Type Conversion

(cast-type) expression

(int) 12.8 -> ?

*(int) 12.8 * 3.2 -> ?*

Basic I/O

- **Output**
- **printf**(format string, var1, var2, …)
  - ➢Format string contains:
    - d: integers
    - f: float, double
    - e: float, double in exponential notation
    - c: character
    - s: string

Basic I/O

- **Input**
- **scanf**(format string, var1, var2, …)
  - – var1, var2, ..: addresses of memory locations!
  - – Format string contains:
    - d,i: integers
    - f: float, double
    - e: float, double in exponential notation
    - c: character
    - s: string

Example

- Printing Integer, Char, using type conversion just when printing
- Printing asci code of a char
- Swapping

Arithmetic & Logical Expressions

- **Arithmetic Operators & Precedence**
- C uses infix notation: a + b * c
- prefix notation: + a * b c
- postfix notation: a b c * +

Operator	Type	Associativity
+ -	Unary	Right to left
* / %	Binary	Left to right
+ -	Binary	Left to right

Arithmetic & Logical Expressions

- **Increment, Decrement Operators**
- ++a, --a

  vs
- a++, a--

Assignment

Compound Assignment Operators

- variable = expression;
    - a = b;
- += -= *= /= %=
    - a += b; -> a = a + b;

Some examples

- i += j = k;
- i = j += k;

```
int i = 7;
int j = 3;
i = j = 5;
print(i,j) = ?
```

Simple Macros

- For long and/or frequent constants:
  - **#define** PI 3.14159265
- For long and/or frequent calculations:
  - **#define** Area(Radius) (4*PI*Radius*Radius)
  - … a = 10.0 + Area(2.0);

Example

- A gasoline (‘benzin’) and diesel engine versions of the same car model consume different amounts of petrol: p_g, p_d (in liters per km), usually $p_g > p_d$. These two different versions of the same car model have different prices: c_g, c_d (usually, $c_g < c_d$).

- Write a program that gets the values p_g, p_d, c_g, c_d as well as the price of 1 liter gasoline and 1 liter diesel from the user and calculates in how many kilometers the price difference these two versions is amortized.

Examples

-
  ```
  main()
  {
  float me = 1.1;
  double you = 1.1;
  if(me==you)
  printf(“Me & You");
  else
  printf(“You & Me");
  }
  ```

-
  ```
  main()
  {
  static int var = 5;
  printf("%d ",var--);
  if(var)
  main();
  }
  ```

Example

-
  ```
  #define square(x) x*x
  main()
  {
  int i;
  i = 64/square(4);
  printf("%d",i);
  }
  ```

Examples

-
  ```
  #include <stdio.h>
  #define a 10
  main()
  {
  #define a 50
  printf("%d",a);
  }
  ```

-
  ```
  void main()
  {
  int i=5;
  printf("%d",i++ + ++i);
  }
  ```

Control Flow

- **Selective Structures**
  - Conditional Expressions and Statements
  - Nested Conditionals
  - Multiway Conditionals
- **Repetitive Structures**
  - While loop
  - Do-while loop
  - For loop
  - Nested loops
  - Loop Interruption(break, continue)

Selective Structures

- **Conditional Expressions and Statements**
  - **Relational** (<, <=, >, >=, ==, !=)
  - **Logical Operators** (&&, ||)
  - **Changing the flow of the program**
    - Conditional statements
    - Conditional expressions

Conditional Expressions and Statements - Relational Operators

- <   <=   >   >=   ==   !=
- False means 0 (zero)
- True means anything that is not False (i.e., non-zero)

Operator	Type	Associativity
+ - ++ --	Unary	Right to left
* / %	Binary	Left to right
+ -	Binary	Left to right
< <= > >=	Binary	Left to right
== !=	Binary	Left to right
= *= /= %= += -=	Binary	Right to left

Conditional Expressions and Statements

Logical Operators

- && || !

Operator	Type	Associativity
+ - ++ -- !	Unary	Right to left
* / %	Binary	Left to right
+ -	Binary	Left to right
< <= > >=	Binary	Left to right
== !=	Binary	Left to right
&&	Binary	Left to right
||	Binary	Left to right
= *= /= %= += -=	Binary	Right to left

Conditional Expressions and Statements

Changing the flow of the program

- if statements

```
if(expr)
{ ….
}
else if(expr)
{…
}
…
else
{ … }
```

```
if(a > b)
       printf(“a is bigger”);
else if(a < b)
        printf(“b is bigger”);
else
        printf(“a = b”);
```

Conditional Expressions and Statements

Changing the flow of the program

- Common mistake with if statements
- **if**(a = 10) { … }
- **if**(a == 10); { … }

Conditional Expressions and Statements

Conditional Expression Operator

- Conditional expression:
  - Expr ? True-expr : False-expr
  - int a = x > 10 ? 1 : 0;
- Right-to-left associative.
  - X = c ? a : d ? e : f;
- Precedence:
  - c ? X = a : X = b
  - ‘?’ and ‘:’ bracket the expression. True-expr can have operators of any precedence without parentheses.
  - The False-expr part has lower precedence than all operators except ‘=‘ and ‘,’.

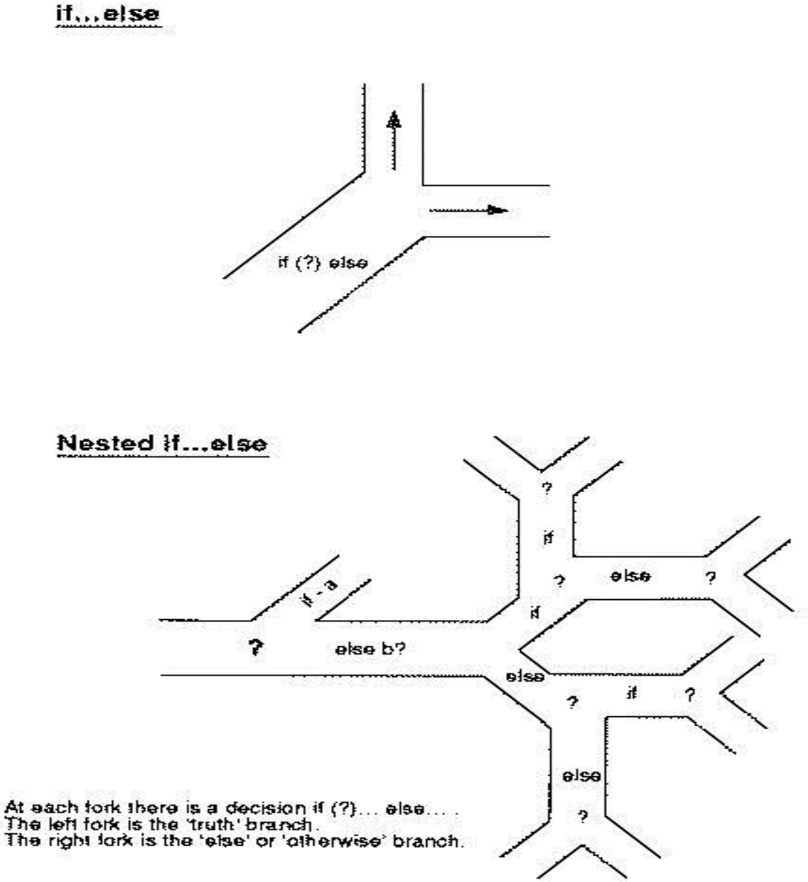
Conditional Expressions and Statements

Nested Conditionals

- if(…)

```
    if( … )
    {….}
else
{….}
```

Conditional Expressions and Statements



At each fork there is a decision if (?)... else....
The left fork is the 'truth' branch.
The right fork is the 'else' or 'otherwise' branch.

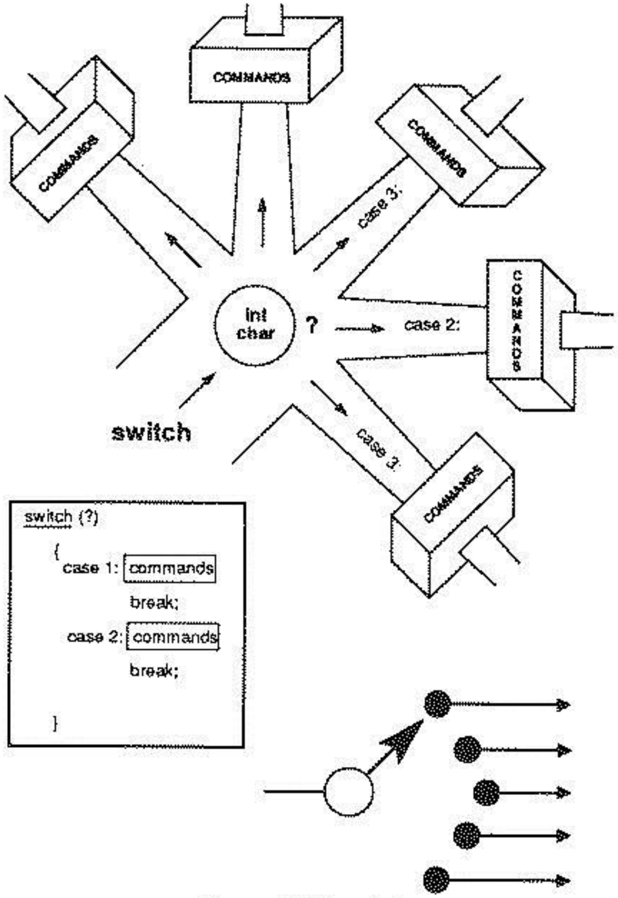
Figure 17.2. Which route – if...else selects.

Conditional Expressions and Statements

- **Multi-way conditionals: switch statements**

```
switch(expr)
{
   case  value-1:
         ….
         break;
   case  value-2:
         ….
         break;
   default:
         ….
         break;
}
```

Conditional Expressions and Statements



```
switch (?)
    {
    case 1: commands
            break;
    case 2: commands
            break;

    }
```

Figure 17.3. switch.

Example

- Making a basic calculator with addition, substraction, etc.

I/O:

5 a 6 -> 11

7 s 4 -> 3

3 m 9 ->27

Example

-
  ```
  main()
  {
  int i=3;
  switch(i)
  {
  default: printf("zero");
  case 1: printf("one");
  break;
  case 2: printf("two");
  break;
  case 3: printf("three");
  break;
  }
  }
  ```

-
  ```
  main()
  {
  int i=1;
  switch(i)
  {
  default: printf("zero");
  case 1: printf("one");
  case 2: printf("two");
  break;
  case 3: printf("three");
  break;
  }
  }
  ```

Example

- Write a C program that classifies a given character into one of the following:
  - Number
  - Uppercase letter
  - Lowercase letter
  - Operator
  - Whitespace

Repetitive Structures

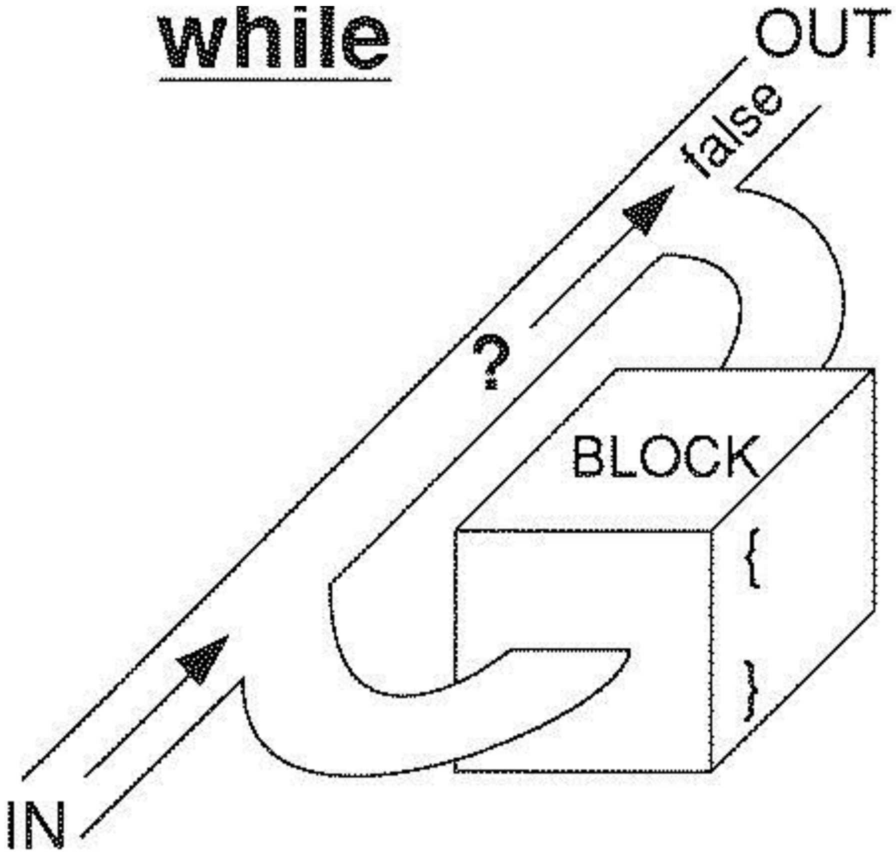


Figure 18.1. The structure of the while command.

Repetitive Structures

while loop

```
Initialization;
while( expr )
    statement;

Initialization;
while( expr )
{
    statement;
    statement;
    statement;
}
```

- Bad examples:

```
while( x = 1)
{
      x = getchar();
}


x = 0.0;
while( x != 1.0 )
{
      x += 0.005;
}
```

Example

Factorial

```
int N, fact = 1;
scanf(“%d”, &N);
while( N > 0 )
{   fact *= N--;  }
```

Repetitive Structures

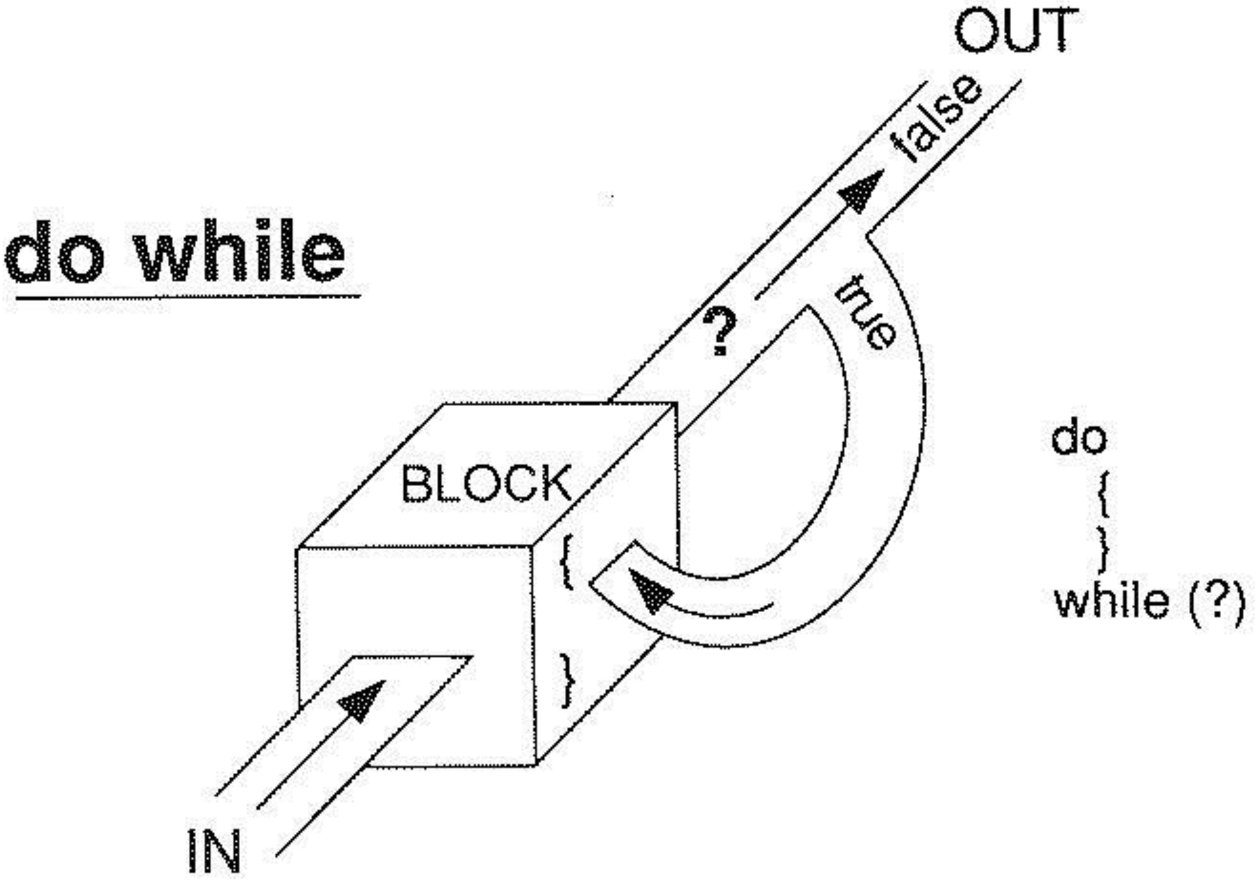


Figure 18.2. The do...while command structure.

Repetitive Structures

do-while loop

```
Initialization;
do
    statement
while( expr );
     statement;

Initialization;
do
{
     statement;
      statement;
      statement;
} while( expr );
```

```
do
{
        x = getchar();
       putchar(x);
} while( x != EOF );
```

Repetitive Structures

- **for** loop

```
Initialization;
for( expr1; expr2; expr3 )
    statement
```

```
Initialization;
for( expr1; expr2; expr3 )
{
        statement;
        statement;
        statement;
}
```

```
for( j = 0; j < N; j++)
        printf(“j: %d\n”, j);
```

```
for(i=0, j=0;
        i < 0 & j > N; i++, j--);
```

```
for(   ;    ; i++ )
{
        if( i > 0 ) return 0;
}
```

Nested Loops

- You can have loops within loops:

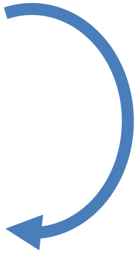
```
for(i=0; i<N; i++)
{
  for(j=0; j<N; j++)
  {
     ….
  }
}
```

Loop Interruption

break;

- Stop the loop/iteration and continue with the statement after the loop.
- Usable with while, for and do-while

```
while(…)
{ …
     break;
  ….
}
statement-X;
```



```
while( 1 )
{
     c = getchar();
     if( c == EOF)
          break;
     putchar( c );
}
```

Loop Interruption

continue;

- Skips the remaining statements in the loop and continues with the “loop head”.
- Usable with while, for and do-while

```
while(…)
{ …
     continue;
  ….
}
```

```
Sum = 0;
for(i=0; i<N; i++)
{
     if( i%2 == 0 )
          continue;
     sum = sum + i;
}
```

Example

- Write a program that gets two number from the user and then prints the numbers between those. If they are equal, warn the user and request again.

Example

- Write a C code that multiplies two numbers without using *, / or %.

Example

- Write a program that reverse a given number.

e.g.

1984->4891

- Write a program that reverse a given string.

e.g.

Hasan -> nasaH