CSE 101
Slide Set 4

Doç. Dr. Mehmet Göktürk
Department of Computer Engineering

1

C Programming Language

- A popular myth about *C is that it has become obsolete and no one is using it anymore*.
- But, it is an inevitable fact that ***C is regarded as one of the oldest and fundamental languages widely used all over the world.***
- The knowledge of programming is incomplete without the incorporation of the C language.
- It continues to dominate the area of programming.

2

Examples of use of C

3

A must have ability..

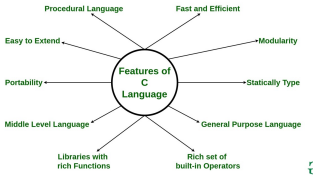
- Many of the world's leading companies are using C programming for their professional use, which clears the fact that C is not an outdated language.
- It is still the most preferred language for programmers and back-end developers.

IF YOU LEARN **C** YOU CAN LEARN OTHERS OVER THE WEEKEND LATER !!

4

C Programming Language

- WHY C ?



5

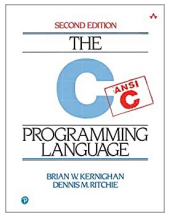
Object Oriented and further expanded C

C ++

6

C Programming Language

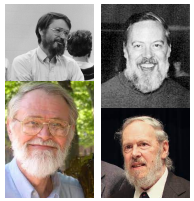
- Introduction
- Statements
- Assignment
- Expressions
- Conditional Expressions
- Loops



7

C Programming Language

- Introduction
- Statements
- Assignment
- Expressions
- Conditional Expressions
- Loops



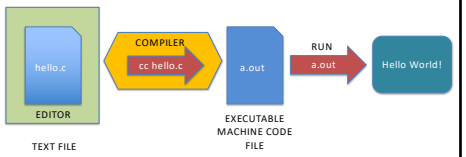
8

Hello World !!

```
#include <stdio.h>
main()
{
   printf("hello, world\n");
}
```


9

How to run?



10

Headers ?

- A lot of predefined things, just include for now..

```
#include <stdio.h>
```

Excercise: check what is inside stdio.h

11

Special characters (escape characters)

```
\n
\\
\t
\b
```


12

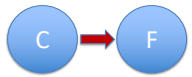
Various ways of writing

```
#include <stdio.h>
main()
{
   printf("hello, ");
   printf("world");
   printf("\n");
}
```


13

Variables and Arithmetic Expressions

- The next program uses the formula °C=(5/9)(°F-32) to print the following table of Fahrenheit temperatures and their centigrade or Celsius equivalents




```
1    -17
20   -6
40   4
60   15
80   26
100  37
120  48
140  60
160  71
180  82
200  93
220  104
240  115
260  126
280  137
300  148
```


14

Program

```
#include <stdio.h>
/* print Fahrenheit-Celsius table
    for fahr = 0, 20, ..., 300 */
main() {
    int fahr, celsius;
    int lower, upper, step;
    lower = 0;  /* lower limit of temperature scale */
    upper = 300;    /* upper limit */
    step = 20;  /* step size */
    fahr = lower;
    while (fahr <= upper) {
        celsius = 5 * (fahr-32) / 9;
        printf("%d\t%d\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```


15

Comment (very important. Why…?)

```
/*  blablabla your explanations  */
```

Or new c++ style as follows on platforms that support c++:

```
// blablabla
```

```
/* this
is comment
*/
```

```
// this
// is
// comment
```


16

Declaration

- In C, all variables must be declared before they are used, usually at the beginning of the function before any executable statements.
- A declaration announces the properties of variables; it consists of a name and a list of **variables**, such as

```
int fahr, celsius;
int lower, upper, step;
```


17

Variables/Numbers: what is a variable?

- The type int means that the variables listed are integers; by contrast with float, which means floating point, i.e., numbers that may have a fractional part
- The range of both int and float depends on the machine you are using; 16-bits ints, which lie between -32768 and +32767, are common, as are 32-bit ints
- A float number is typically a 32-bit quantity, with at least six significant digits and magnitude generally between about 10^{-38} and 10^{38}

18

Common Data Types in C

int - integer
float - floating point number
char character - a single byte
short - short integer
long - long integer
double - double-precision floating point

The size of these objects is also machine-dependent. There are also **array**s, **structure**s and **union**s of these basic types, **pointer**s to them, and **function**s that return them

19

Assignment

```
lower = 0;
upper = 300;
step = 20;

lower = 0; upper = 300; step = 20;

i=j=k=0;
```

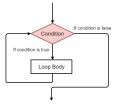
WHY NOT `20=step;` ??

What is `;`

20

while loop

```
while (fahr <= upper) {
    ...
}
```



- The condition in parentheses is tested.
- If it is true (fahr is less than or equal to upper), the body of the loop (the three statements enclosed in braces) is executed.
- Then the condition is re-tested, and if true, the body is executed again. When the test becomes false (fahr exceeds upper) the loop ends, and execution continues at the statement that follows the loop.
- There are no further statements in this program, so it terminates.

21

`while` loop

- The body of a while can be one or more statements enclosed in braces, as in the temperature converter, or a single statement without braces, as in

```
while (i < j)
    i = 2 * i;
```

```
while( ..... ){
    ...
    ...
    ...
}
```

```
while(1==1);
```

?????

22

`printf` (output..)

```
printf("%d\t%d\n", fahr, celsius);
```

values of the two integers fahr and celsius to be printed, with a tab (\t) between them.

```
1     -17
20    -6
40    4
60    15
80    26
100   37
120   48
140   60
160   71
180   82
200   93
220   104
240   115
260   126
280   137
300   148
```


23

Formatted output

```
printf("%3d %6d\n", fahr, celsius);
```

```
  0     -17
 20      -6
 40       4
 60      15
 80      26
100      37
...
```

Formatted print the first number of each line in a field three digits wide, and the second in a field six digits wide

24

Better version

```
#include <stdio.h>

/* print Fahrenheit-Celsius table
   for fahr = 0, 20, ..., 300; floating-point version */
main()
{
    float fahr, celsius;
    int lower, upper, step;

    lower = 0;      /* lower limit of temperature table */
    upper = 300;    /* upper limit */
    step = 20;      /* step size */

    fahr = lower;
    while (fahr <= upper) {
        celsius = (5.0/9.0) * (fahr-32.0);
        printf("%3.0f %6.1f\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```

25

Formatted output ….

- Width and precision may be omitted from a specification: %6f says that the number is to be at least six characters wide; %.2f specifies two characters after the decimal point, but the width is not constrained; and %f merely says to print the number as floating point.

`%d`	print as decimal integer
`%6d`	print as decimal integer, at least 6 characters wide
`%f`	print as floating point
`%6f`	print as floating point, at least 6 characters wide
`%.2f`	print as floating point, 2 characters after decimal point
`%6.2f`	print as floating point, at least 6 wide and 2 after decimal point

Excercise: Write a program to print the corresponding Celsius to Fahrenheit table

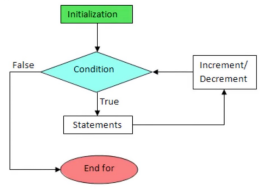
26

`for` loop

```
#include <stdio.h>

/* print Fahrenheit-Celsius table */
main()
{
    int fahr;

    for (fahr = 0; fahr <= 300; fahr = fahr + 20)
        printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
}
```

27

`for` loop




```
for(i=0;i<10;i++){
    printf("Hello\n");
}
```

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
hello
```

28

Symbolic Constants : `#define`

```
#include <stdio.h>

#define   LOWER  0       /* lower limit of table */
#define   UPPER  300     /* upper limit */
#define   STEP   20      /* step size */

/* print Fahrenheit-Celsius table */
main()
{
    int fahr;

    for (fahr = LOWER; fahr <= UPPER; fahr = fahr + STEP)
        printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
}
```


29

Character Input and Output

- The standard library provides several functions for reading or writing one character at a time, of which getchar and putchar are the simplest.
- Each time it is called, getchar reads the next input character from a text stream and returns that as its value.

```
c = getchar()
```

- the variable c contains the next character of input. The characters normally come from the keyboard; input from files is discussed later

```
putchar(c)
```

- prints the contents of the integer variable c as a character, usually on the screen. Calls to putchar and printf may be interleaved; the output will appear in the order in which the calls are made.

30

File Copying (file and stream principle)

- Given getchar and putchar, you can write a surprising amount of useful code without knowing anything more about input and output.
- The simplest example is a program that copies its input to its output one character at a time

```
read a character
while (character is not end-of-file indicator)
    output the character just read
    read a character
```


31

C Program

```
#include <stdio.h>

/* copy input to output; 1st version */
main()
{
    int c;

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

EOF ???

!= means "not equal to."

32

`char` and `int`

- The problem is distinguishing the end of input from valid data.
- The solution is that `getchar` returns a distinctive value when there is no more input, a value that cannot be confused with any real character.
- This value is called **EOF**, for ``**end of file**''.
- We must declare **c** to be a type big enough to hold any value that `getchar` returns.
- We can't use `char` since **c** must be big enough to hold **EOF** in addition to any possible char. Therefore we use `int`.
- **EOF** is an integer defined in **<stdio.h>,**

33

Second version

```
#include <stdio.h>

/* copy input to output; 2nd version */
main()
{
    int c;

    while ((c = getchar()) != EOF)
        putchar(c);
}
```

The parentheses around the assignment, within the condition are necessary !!

34

Character Counting

```
#include <stdio.h>

/* count characters in input; 1st version */
main()
{
    long nc;

    nc = 0;
    while (getchar() != EOF)
        ++nc;
    printf("%ld\n", nc);
}
```

`++nc;`

35

Size of count??

- The character counting program accumulates its count in a long variable instead of an int. long integers are at least 32 bits.
- Although on some machines, int and long are the same size, on others an int is 16 bits, with a maximum value of 32767, and it would take relatively little input to overflow an int counter.
- The conversion specification %ld tells printf that the corresponding argument is a long integer.
- It may be possible to cope with even bigger numbers by using a double (double precision float).

36

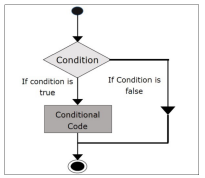
For loop version

```
#include <stdio.h>

/* count characters in input; 2nd version */
main()
{
    double nc;

    for (nc = 0; getchar() != EOF; ++nc)
        ;
    printf("%.0f\n", nc);
}
```


37

Conditional Expressions

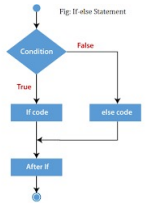
```
if( ..... ){
   bla bla bla
   bla bla bla
}
```



38

Conditional Expressions

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



39

Conditional Cascaded

```
if( ..... ){
   bla bla bla
   if( ..... ){
      bom bom
   }
   blu blu blu
}
else{
   if ( ..... ){
      bla bla bla
      bla bla blom
   }
   else{
      bom
   }
   bambum
}
```


40

Conditional Cascaded

```
if( ..... ){
    bla bla bla
    if( ..... ){
        bom bom
    }
    blu blu blu
}
else if( ..... ){
        bla bla bla
        bla bla blom
}
else{
    bom
}
```

bambum?

41

Line Counting

```
#include <stdio.h>

/* count lines in input */
main()
{
    int c, nl;

    nl = 0;
    while ((c = getchar()) != EOF)
        if (c == '\n')
            ++nl;
    printf("%d\n", nl);
}
```

42

Word Counting

```
#include <stdio.h>

#define IN  1   /* inside a word */
#define OUT 0   /* outside a word */

/* count lines, words, and characters in input */
main()
{
    int c, nl, nw, nc, state;

    state = OUT;
    nl = nw = nc = 0;
    while ((c = getchar()) != EOF) {
        ++nc;
        if (c == '\n')
            ++nl;
        if (c == ' ' || c == '\n' || c == '\t')
            state = OUT;
        else if (state == OUT) {
            state = IN;
            ++nw;
        }
    }
    printf("%d %d %d\n", nl, nw, nc);
}
```

43

Pay attention to:

```
nl = nw = nc = 0;
```

```
nl = (nw = (nc = 0));
```

```
if (c == ' ' || c == '\n' || c == '\t')
```

44

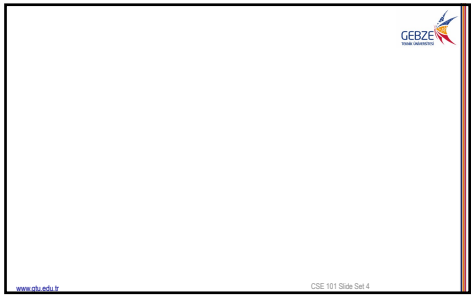
Examples..

45

46

47

48



49