Computers

- A **computer** is a machine capable of following the instructions of a **program.**

- A **program** is a set of instructions.

- **Computer Organization: Hardware**
  - The Central Processing Unit: (CPU)
    1. Carries out the instructions of the programs.
    2. Moves data from one part of machine to another part of the machine.
    3. Manipulates data (e.g., adds, compares, etc.).

  - Memory:
    1. Internal Memory: memory internal to the system printed circuit boards. Used to store programs and data that are currently being processed by the CPU.
       a) RAM - Random Access Memory
       b) ROM - Read Only Memory
    2. External Memory: memory external to the system printed circuit boards. Used for long term and mass storage of programs and data.
       a) Floppy Disks
       b) Hard Disks
       c) Magnetic Tape
       d) CD ROM

  - Input/Output Devices:
    1. Input Devices: used to input programs and data into the computer.
       a) Keyboard
       b) Mouse or Trackball
    2. Output Devices: used to receive output information from the computer.
       a) Screen
       b) Printer

- **Computer Organization: Software**

  - Program:
    A set of instructions that determines a computer's action.

  - Software:
    A program or a collection of programs that are not 'built-in' to the hardware.

  - The Operating System:
    The software that manages the entire computer system.

  - Programming Language:
    A language with which programs are written.

  - C, C++, Java, Pascal:
    High-level programming languages.

  - Assembly Language:
    A low-level programming language.

  - Machine Language
    The "language" that the computer actually understands. (strings of 1s and 0s)

  - Compiler:
    Software that translates programs written in high-level languages into machine language.

Simple C Programs

- Example:

```
/*  display a message on the monitor * /
#include < stdio.h>
void main()
{
   printf("Welcome to computer programming!\n");
}
```

- Example:

```
/*  type in a number on the keyboard & echo it to the screen* /
#include < stdio.h>
void main()
{
   int number;

   printf("Type in a number on the keyboard\n");
   scanf("%d",&number);
   printf("The number is %d\n",number);
}
```

- Example:

```
/*  a program to do some addition * /
#include < stdio.h>
void main()
{
   int number;

   printf("Type in a number on the keyboard\n");
   scanf("%d",&number);
   number =  number +  5;
   printf("The new  number is %d\n",number);
}
```

The C Programming Language

- Like any language, C has:

  - A **Lexicon** (a dictionary):
    A set of allowable basic elements.

  - A **Syntax** (a grammar):
    Rules for combining the basic elements into programs.

  - A **Semantics** (meaning):
    Rules for understanding what programs and their parts cause the computer to do.

- Sample Program:

```
/*  a sample C program to add some numbers */
#include < stdio.h>
void main()
{
   int number1, number2, sum;

   printf("Type in the first number\n");
   scanf("%d",&number1);
   printf("Type in the second number\n");
   scanf("%d",&number2);
   sum =  number1 +  number2 +  5;
   printf("The final sum is %d",sum);
}
```

The Lexical Elements of C

- **Keywords:**
  Reserved Words or Predeclared Identifiers that have special meaning in C:

- main	- char	- printf	- for	- if	- switch
- int	- float	- scanf	- while	- else	- struct

- **Identifiers:**
  Words that the writer of the program chooses to name things (e.g., functions, variables, etc.).
  Identifiers must start with a letter or an underbar. The remaining characters are any combination of letters, numerals, and underbars. (Note: upper and lower case are different.)
  - number1
  - Number1
  - sum
  - R2D2
  - Last_Name

- **Strings:**
  Anything that is written between double quotes.
  - "The final sum is "
  - "hello!"

- **Numeric Constant:**
  Numerals which express specific numeric values.
  5
  227
  33.676

- **Special Characters:**
  Characters or groups of characters that have a special meaning in C.
  ; , { } < <= > >= == + - * % & && | || etc.

The Program Syntax of C

- Rules for grouping C statements into C programs.

- **The Structure of a Simple C Program:**

```
/*  comment on purpose of program */  // comment
#directives                          // compiler directives
void main()                          // program header
{                                    // start of program
   data_type list_of_identifiers;    // variable declarations
   < data_type1>  < list1> ;
   < data_type2>  < list2> ;
...
   action part of the program        // do program's task
}                                    // end of program
```

- A **variable** is a part (location) of memory that is capable of holding a piece of data that may change its value.

- **The 'variable declaration' Part of the Program:**
  Indicates how much space in memory that the action part of the program will need, and the identifiers that will be used to refer to those locations. These identifiers are called the variables of the program.

- **The Action Part of the Program:**
  The series of C statements (between "{" and "}") that accomplish the program's task.

The Statement Syntax of C

- Rules for combining lexical elements into C statements.

- **The Declaration Statement Syntax:**
  data_type variable_1,...,variable_n;

```
int number;
int num1, num2, num3;

integer number;             // illegal
ints num1, num2, num3;      // illegal
int switch;                 // illegal
```

- **C Expressions:**

- an integer constant	3
- an identifier	x
- an identifier, an operator, an integer constant	sum + 10
- an integer constant, an operator, an identifier	10 * y
- an identifier, an operator, an identifier	sum - num

- **The Assignment Statement Syntax:**
  variable = *expression*;

```
z = 33;
x = y;
sum = number1 + number2 - 5;
sqnumber = number * number;
```

The printf Statement

- **The printf Statement Syntax:**
  printf("*format string*", *item1,item2,...,itemN*);

  where each item is either a **variable** or an **expression** .

- The printf always contains a **format (or control) string**.
- The list of items is optional.
- Each item to be printed needs a **conversion specification** within the format string. The conversion specification describes the exact way that the item is to be printed.

- Example:
```
number = 4;
sqnumber = number * number;
printf("The square of %d is %d\n",number,sqnumber);
```

- The **%d** specification means to print as a decimal number (i.e., an integer).
- The **\n** means **newline** and tells the computer to skip to the next line.

- The above code prints:
```
The square of 4 is 16
```

- Another Example:
```
printf("%d\n",number+1);
```

C Statement Types

- C statements can be simple or compound.
- **Simple Statement (Examples):**

```
printf("hello!");

x =  25;
```

- **Compound Statement:**

```
{
    statement-list
}
```

- **Statement-List:**
  A list of C statements. Each statement may be simple or compound.

```
    statement;
    statement;
    ...
    statement;
    statement;
```

Arithmetic Operations

- Addition:
  ```
  a +  b
  ```
- Subtraction:
  ```
  a - b
  ```
- Multiplication:
  ```
  a *  b
  ```
- Division:
  ```
  a / b

  a %  b              (yields remainder of integer division)
  ```
- Examples:
  ```
  int number,sqnumber;
  int num1,num2;
  int quotient,remainder;

  number =  4;
  sqnumber =  number *  number;    {sqnumber is assigned 16}

  num1 =  5;
  num2 =  37;
  quotient =  num2 / num1;         {quotient is assigned 7}
  remainder =  num2 % num1;        {remainder is assigned 2}
  ```

Conditions

- A C **condition** is a Boolean expression of the form:
  expression_1 relational_operator expression_2
  which takes on the value of either TRUE or FALSE.

- **Relational Operators:**

<	less than
< =	less than or equal to
= =	equal to
!=	not equal to
>	greater than
> =	greater than or equal to

- Example:
  Let: n = 15; sum = 11

  Then:

sum < n	TRUE
n < = 0	FALSE
n > 0	TRUE
n > = sum - 1	TRUE
n = = n + 1	FALSE
n + 2 != 17	FALSE

A Complete C Program

- **Problem:**
  Write a C program to compute and print the squares of the integers from 4 to 8.

- **Program:**

```
/*  program to print the number from 4 to 8 and their squares */
   #include < stdio.h>
   void main()
   {
      int number,sqnum;

      number =  4;
      sqnum =  number *  number;
      printf("number =  %d sqnum =  %d\n",number,sqnum);
      number =  5;
      sqnum =  number *  number;
      printf("number =  %d sqnum =  %d\n",number,sqnum);
      number =  6;
      sqnum =  number *  number;
      printf("number =  %d sqnum =  %d\n",number,sqnum);
      number =  7;
      sqnum =  number *  number;
      printf("number =  %d sqnum =  %d\n",number,sqnum);
      number =  8;
      sqnum =  number *  number;
      printf("number =  %d sqnum =  %d\n",number,sqnum);
   }
```

- **Output of the Program:**

```
number =  4 sqnum =  16
number =  5 sqnum =  25
number =  6 sqnum =  36
number =  7 sqnum =  49
number =  8 sqnum =  64
```

Iteration

- **Repeating a Series of Instructions:**

```
number = 4;
sqnum = number * number;
printf("number = %d sqnum = %d\n",number,sqnum);
number = 5;
sqnum = number * number;
printf("number = %d sqnum = %d\n",number,sqnum);
number = 6;
sqnum = number * number;
printf("number = %d sqnum = %d\n",number,sqnum);
```

- The **for statement** (**or for loop**) repeats a group of statements while a control-variable is between its initial and final values.

- **The for Statement:**

  for (*exp1; exp2; exp3*)
      body (statement) of the loop;

  where
  exp1 - initializes one or more variables,
  exp2 - performs a relational test,
  exp3 - updates one or more variables.
  body of loop can be either a simple statement or a compound statement enclosed with braces.

- Example:

```
for (j = 1; j <= 10; j = j+1)
   printf("%d\n",j);
```

- Example:

```
for (number = 4; number <= 6; number = number+1) {
   sqnum = number * number;
   printf("number = %d sqnum = %d\n",number,sqnum);
}
```

- Example:

```
for (number =  8; number > =  4; number =  number-1) {
    sqnum =  number *  number;
    printf("number =  %d sqnum =  %d\n",number,sqnum);
}
```

- **Revised Program**

- **Problem:**
  Write a C program to compute and print the squares of the integers from 4 to 8.

- **Program:**

```
/*  program to print the number from 4 to 8 and their squares */
   #include < stdio.h>
   void main()
   {
       int number,sqnum;

       for (number =  4; number < =  8; number= number+ 1) {
           sqnum =  number *  number;
           printf("number =  %d sqnum =  %d\n",number,sqnum);
       }
   }
```

- **Trace of the Program**

```
number =  4 sqnum =  16
number =  5 sqnum =  25
number =  6 sqnum =  36
number =  7 sqnum =  49
number =  8 sqnum =  64
```

Shorthand for Increments and Decrements

- Because increments and decrements are so common, C has a shorthand notation for these operations:

  j = j + 1; can be replaced by j++;

  i = i - 1; can be replaced by i--;

- Example: Use in for loops

```
for (number = 4; number <= 8; number++) {
    sqnum = number * number;
    printf("number = %d sqnum = %d\n",number,sqnum);
}
```

- Example:

```
for (number = 8; number >= 4; number--) {
    sqnum = number * number;
    printf("number = %d sqnum = %d\n",number,sqnum);
}
```

- Example

```
for (i = 1, j = 7; i < j; i= i+ 3, j++)
    printf("%d %d\n",i,j);
```

  - Output:

```
1 7
4 8
7 9
```

- Note: ++j will also increment (can be different than j++)
- Note: --i will also decrement (can be different than i--)