Fundamentals of Computer and Programming

Lecture 4

Calculations

Instructor: Morteza Zakeri, Ph.D.

(zakeri@aut.ac.ir)

Modified Slides from Dr. Hossein Zeinali and Dr. Bahador Bakhshi

School of Computer Engineering,

Amirkabir University of Technology

Spring 2025

What We Will Learn

- Basic mathematic operations in C
- Effect of type and type conversion
- Precedence
- Advanced mathematical operations
- Mathematic library
  - Random numbers

Basic operations

عملگر (operator)	مفهوم محاسباتی
+	جمع
-	تفریق
/	تقسیم
*	ضرب
%	باقیمانده

Example

1 + 2 → 3

1 + 2 + 3 + 4 → 3 + 3 + 4

→ 6 + 4

→ 10

10 * 20 → 200

100 / 20 → 5

Modulo

- %
- Only can be used by **`int`** **operands**

5 % 4 → 1

7 % 88 → 7

-20 % 7 → -6

20 % -7 → 6

-20 % -7 → -6

Parenthesis

(2 + 5) * (7 – 1) → (7) * (6) → 42

1 * (2 + (3 * (4 + 5))) → 1 * (2 + (3 * (9)))
→ 1 * (2 + (27))
→ 1 * (29)
→ 29

(((1 * 2) + 3) * 4) + 5 → (((2) + 3) * 4) + 5
→ ((5) * 4) + 5
→ (20) + 5
→ 25

برنامه چاپ میانگین سه عدد

```c
#include <stdio.h>

int main(void){
        float num1, num2, num3, sum, average;
        printf("Enter 3 number: \n");
        scanf("%f",&num1);
        scanf("%f",&num2);
        scanf("%f",&num3);
        sum = num1 + num2 + num3;
        average = sum / 3;
        printf("Miangin = ");
        printf("%f\n", average);
        return 0;
}
```

What We Will Learn

- Basic mathematic operations in C
- Effect of type and type conversion
- Precedence
- Advanced mathematical operations
- Mathematic library
  - Random numbers

General rules of type conversion

- If either operand is long double, convert the other to long double.
- Otherwise, if either operand is double, convert the other to double.
- Otherwise, if either operand is float, convert the other to float.
- Otherwise, convert char and short to int.
- Then, if either operand is long, convert the other to long.

Effect of types

- Type of operands determines the type of the result
  - The type of output is the type of operands (after conversion)
- int <op> int → int
- int <op> long → long
- float <op> float → float
- float <op> int → float
- double <op> float → double

(a) `5 + 2.0` ⇒ `7.0`	The result is a `double`.
(b) `3 * 4L` ⇒ `12L`	The result is a `long`.
(c) `2.5f + 2.5` ⇒ `5.0`	The result is a `double`.

Effect of types

- If both operand of division (/) is int
  - → data lost

(a) 15/3 ⇒ 5

(b) 13/4 ⇒ 3

(c) 9/5 ⇒ 1

(d) 7/9 ⇒ 0

(e) 27L/10L ⇒ 2L

(f) 9999L/10000L ⇒ 0L

(g) 7/(-3) ⇒ -2

(h) -15/4 ⇒ -3

(i) -5/(-6) ⇒ 0

(j) (-9)/(-5) ⇒ 1

Effect of types & Explicit casts

Expression		*Type of result*
(double) 1 + 2.0f	→ 3.0	double
(int) 2.69 + 4	→ 6	int
(double) 1 / 2	→ 0.5	double
1 / (int) 2.0	→ 0	int
(double) (1 / 2)	→ 0.0	double
(int)((double) 1 / 2)	→ 0	int

What We Will Learn

- Basic mathematic operations in C
- Effect of type and type conversion
- Precedence
- Advanced mathematical operations
- Mathematic library
  - Random numbers

Precedence (اولویت)

- 1) Parenthesis
- 2) Unary + - (for sign): +4, -8
- 3) Explicit casting
- 4) / * %
- 5) Binary + -: 4+8
- 6) If multiple + - or / * %: from left to right

-5 + 2 / 4.0 * (-7 / 8) → -5 + 2 / 4.0 * (0)

→ -5 + 0.5 * 0

→ -5 + 0.0

→ -5.0

Precedence

(7 + (float) (2 + (int) 1.005)) / (int) 20 →

(7 + (float) (2 + 1)) / (int) 20 →

(7 + (float) (3)) / (int) 20 →

(7 + 3.0f) / (int) 20 →

10.0f / (int) 20 → 0.5 // Result is float

5 + (double)(7 / (int) 8.5 / 7.0 * 6) →

5 + (double)(7 / 8 / 7.0 * 6) →

5 + (double)(0 / 7.0 * 6) →

5 + (double)(0.0 * 6) → 5 + 0.0 → 5.0 // Result is double

برنامه چاپ جمع قسمت صحیح دو عدد اعشاری

```
#include <stdio.h>

int main(void){
        float num1, num2; // ورودی‌ها
        int sum; // حاصل جمع
        printf("Enter 2 number: \n");
        scanf("%f",&num1);
        scanf("%f",&num2);
        sum = (int)num1 + (int)num2;
        printf("%d\n", sum);
        return 0;
}
```

برنامه چاپ جمع قسمت اعشاری دو عدد اعشاری

```
#include <stdio.h>
int main(void){
        float num1, num2, fpart1, fpart2, sum;
        printf("Enter 2 number: \n");
        scanf("%f",&num1);
        scanf("%f",&num2);
        fpart1 = num1 - (int)num1;
        fpart2 = num2 - (int)num2;
        sum = fpart1 + fpart2;
        printf("%f\n", sum);
        return 0;
}
```

What We Will Learn

- Basic mathematic operations in C
- Effect of type and type conversion
- Precedence
- Advanced mathematical operations
- Mathematic library
  - Random numbers

Increment and Decrement of Variables

- Unary operators only for variables
- ++ : increase by one
- -- : decrease by one

```
int i = 10;

i = i + 1; // i = 11
i++;       // i = 12
++i;       // i = 13
i--;       // i = 12
--i;       // i = 11
i = i - 1; // i = 10
```

Increment and Decrement (Cont'd.)

- Postfix: Use the value then apply the operator
- Prefix: Apply the operator then use the value

```
int i = 10, j;
j = i + 1;          // i = 10, j = 11
j = i++;            // i = 11, j = 10
j = ++i;            // i = 12, j = 12
j = i--;            // i = 11, j = 12
j = --i;            // i = 10, j = 10
j = i - 1;          // i = 10, j = 9
```

Assignment Combined with Operation

- These are equal
  - <variable> <op>= <expression>
  - <variable> = <variable> <op> (<expression>)

```
int i = 9, j = 20;

i += 1;          // i = i + 1;    // i = 10
j /= i;          // j = j / i;    // j = 2

i  *= i + j - 6 + i / j;
/* i = i * (i + j - 6 + (i / j)); i = 110 */
```

Multiple assignment

- More than one **assignment** in a statement
  - From right to left

```
int i, j, k, l;
i = j = k = l = 1;
i += j *= --k -= 3 / l;
/* → i += j *= --k -= 3
   → i += j *= --(k -= 3) [k = -2]
   → i += j *= --k [k = -3]
   → i += j *= -3 [j = -3]
   → i += -3 [i = -2]
   i = -2, j = -3, k = -3, l = 1
*/
```

Precedence

Operator	Direction
`( )`	
`- + ++ -- (type)`	
`* / %`	Left to right
`+ -  add. sub.`	Left to right
`= += -= *= /= %=`	Right to left

Arithmetic on characters

- `char` can be used as **8-bit integer**
- All arithmetic operations can be used with characters

```
/* A: 65, B: 66, C: 67, … */
char c = 'A', ch;
int i;
c++;          // c = 66, c = 'B'
ch = c;
ch += 3;      // ch = 69, ch = 'E'
i = c - ch + 'X' - 'Z'; // i = -5
```

sizeof operator

- **sizeof** is a unary operator
  - Return the size of operand
  - Operand can be
    - Variable, value or type

```
int size, i = 10;

size = sizeof i;

size = sizeof(i);

size = sizeof(2000);

size = sizeof(char)
```

`sizeof` operator

- According to C99 Standards, the **`sizeof()`** operator only considers the operand type.
  - may be an expression or the name of a type (*i.e.*, int, double, float, etc.) and not the value obtained on evaluating the expression.
  - The operand inside the **`sizeof()`**operator is not evaluated.
- It is evaluated only if the type of the operand is a variable length array.
  - in that case, the size can be determined only after the expression is evaluated.

Precedence

Operator	Direction
`( )`	
`- + ++ -- (type) sizeof`	
`* / %`	Left to right
`+ -  add. sub.`	Left to right
`= += -= *= /= %=`	Right to left

Complicated examples

```
int i, j, k, n;

i = j = k = n = 1;

i = sizeof(int) + sizeof(char) + sizeof 10;
                                    //i:9

i = j = k = n = 1;

i += j * k++ + sizeof n;
                                    //i:6 j:1 k:2 n:1

i = j = k = n = 2;

i = j + (k = ++n);                  //i:5 j:2 k:3 n:3
```

Undefined Statements

- When the standard does not tell what will happen
- Linux GCC and Code::Blocks outputs are different
- Examples

```
int i, j, k;

k = i = 10;
j = i++ + k + --i;      //j = 29 or 30?

i = j = 10;
i = j + i++;            //i = 11 or 20?
```

Recap: Overflow and Underflow

- Computer's precision is limited
  - The number of bits in each type is limited
  - double [-1e308, 1e308]

- Overflow
  - When result is larger than specified ranges

    1e300 * 1e200

- Underflow
  - When the result is too smaller than precision

    1e-300 * 1e-200

برنامه محاسبه مقدار چند جمله‌ای درجه ۲

```
#include <stdio.h>
int main(void){
        float a, b, c, x, result;
        printf("Enter a, b, c, x: ");
        scanf("%f", &a);
        scanf("%f", &b);
        scanf("%f", &c);
        scanf("%f", &x);
        result = a * x * x + b * x + c;
        printf("%f\n", result);
         return 0;
}
```

What We Will Learn

- Basic mathematic operations in C
- Effect of type and type conversion
- Precedence
- Advanced mathematical operations
- Mathematic library
  - Random numbers

Math Library

- `#include <math.h>`

```
double f = 36;
fabs(-f)                36.000000
sqrt(f)         6.00000
pow(f, 0.5)             6.000000
ceil(-10.2)             -10.000000
ceil(10.2)              11.000000
floor(-10.2)            -11.000000
floor(10.2)             10.000000
fmax(10.1, 20.2)        20.2
fmin(10.1, 20.2)        10.1
rint(10.2)              10.0            rint(-10.2)             -10.0
rint(20.6)              21              rint(-20.6)             -21
```

Math Library

```
const double PI = 3.141592653589793;
const double E = 2.7182818284590451;
```

`sin(PI)`	`0.000000`
`cos(PI/2)`	`0.000000`
`acos(1)`	`0.000000`
`log(E)`	`1.000000`
`log(10)`	`2.30258`
`exp(1)`	`2.718282`

برنامه محاسبه محیط و مساحت دایره

```c
#include <stdio.h>
#include <math.h>
#define PI 3.141592653589793

int main(void){
    float r;
    printf("Enter shoa");
    scanf("%f", &r);
    double masahat = PI * pow(r, 2);
    double mohit = 2 * PI * r;
    printf("masahat = %f\n", masahat);
    printf("mohit = %f\n", mohit);
    return 0;
}
```

برنامه حل معادله درجه دو (با فرض وجود ریشه)

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

int main(void){
        float a, b, c, delta, root1, root2;
        printf("Enter a, b, c: ");
        scanf("%f", &a);
        scanf("%f", &b);
        scanf("%f", &c);
```

برنامه حل معادله درجه دو (با فرض وجود ریشه)

```
        delta = sqrt((b * b) - (4 * a * c));
        root1 = (-b + delta) / (2 * a);
        root2 = (-b - delta) / (2 * a);
        printf("root1 = ");
        printf("%f\n", root1);
        printf("root2 = ");
        printf("%f\n", root2);
         return 0;
}
```

Random Numbers

- `#include <stdlib.h>`
- `rand();`
  - A random number in [0, RAND_MAX]
- How does it work
  - Start from a seed number
    - X0 ← F(seed number)
  - Xn+1 = F(Xn)
- Same seed
  - Same random number sequence

Random Numbers

- We usually want different random number
  - Run 1: 10, 20, 17, 1000, 23, 345, 30
  - Run 2: 23, 904, 23, 346, 85, 234, 63
- We should use different seed in each run
  - How?
  - Initialize seed by system time

```
#include <time.h>
time_t t = time(NULL);
srand(t);
```

Random Numbers

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

int main(void){
   int r1, r2;
   srand(0);
   r1 = rand();
   printf("r1 = %d\n", r1);
   time_t t = time(NULL);
   srand(t);
   r2 = rand();
   printf("r2 = %d\n", r2);
   return 0;
}
```

First Run
r1 = 38
r2 = 1873

Second Run
r1 = 38
r2 = 1866

Third Run
r1 = 38
r2 = 1860

برنامه چاپ یک عدد اعشاری تصادفی در بازه (0, 1)

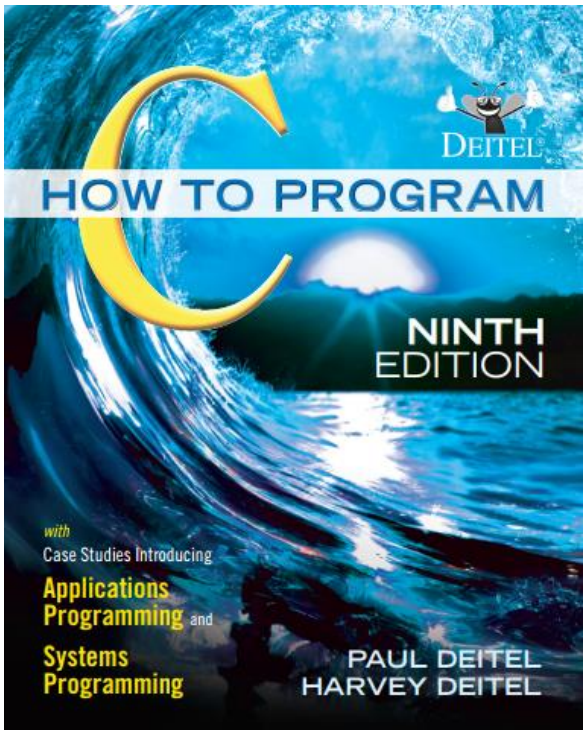
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void){
    time_t t = time(NULL);
    srand(t);

    int ir = rand();
    double fr = (ir + 1) / (RAND_MAX + 2.0);
    printf("%f\n", fr);

    return 0;
}
```

Print 3 random numbers between 0 and 49

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main (){
  int i, n;
  time_t t;
  /* Initializes random number generator */
  srand((unsigned) time(&t));
  /* Print 3 random numbers from 0 to 49 */
  printf("%d\n", rand() % 50);    printf("%d\n",
  rand() % 50);    printf("%d\n", rand() % 50);
  return(0);
}
```


Reference

- Reading Assignment: Sections **2.3** to **2.6** of **Chapter 2** of "C How to Program" and Section and Sections **3.11** to **3.12** of **Chapter 3.**



Questions

- Given the following code snippet, what will be printed?

```
int main() {
int a = 5, b = 2, c = 4;
int result = a + b * c - b % c;
printf("Result: %d\n", result); 15
return 0;
}
```

A) 12 B)11 C)14 D)13

- Answer: A

Questions

- What is the final value of a and b in the code?

```
int main() {
int a = 5, b = 10;
a = b--;
b = ++a;
printf("a = %d, b = %d\n", a, b);
return 0;
}
```

A) a=5, b=9 B) a=11, b=11

C) a=9, b=10 D) a=10, b=10

- Answer: B

Questions

- What is the result of the following code if c is assigned the value INT_MAX + 1?

```
int main() {
int c = INT_MAX + 1;
printf("c = %d\n", c);
return 0;
}
```

A) c = INT_MAX B) c = INT_MIN

C) Undefined behavior D) c = 0

- Answer: B