Week 2: Inputs, Expressions, and Decisions

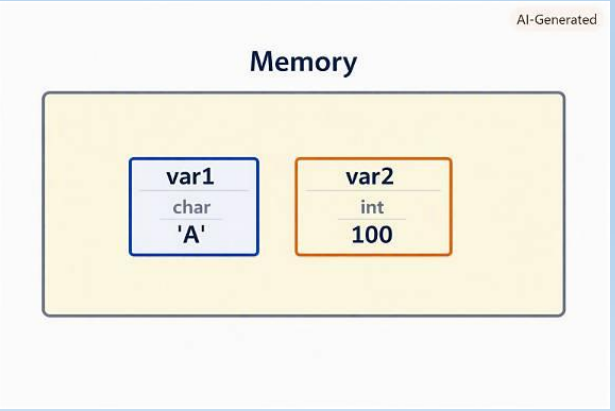
ID1063 • Introduction to Programming in C

Instructor: Rakesh Venkat

Recap from week 1

1. Source Code->Compile->Execute process
2. Starting C program
3. printf statement
    1. Syntax
    2. Special characters, escape sequence

4. Comments in programs
5. Variables
    - Type and Name
    - Declaration
    - Assignment



Printing variables using printf

```
int year = 2026;
printf("This course begins in %d.\n", year);
```

Type	printf placeholder
int	%d
char	%c
float	%f
double	%f

Volume of a box : printing expressions

- Program to evaluate volume of a box and print it

```
#include <stdio.h>

int main(void)
{
    int length, width, height; //integer variables declared
    int volume;
    length=10;
    width=20;
    height=10;

    volume = length*width*height;

    printf("Volume = %d\n", volume);
    return 0;
}
```

- Can use expressions directly within printf

```
printf(“Sum = %d \n”, a+b);
```

- How to reuse for a separate box instance?

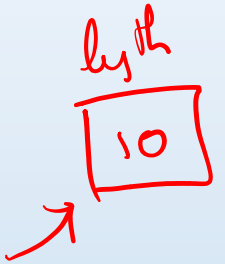
Reading Input

The scanf() function

Reusing programs

- Input: obtain the values needed by the program from user when program is running.
- Computation: evaluate expressions using those values.
- Output: report the result to the user.
- A new run can use different input without changing the source code.

The scanf function




```
int length;

scanf("%d", &length);

printf("You entered %d\n", length);
```

%d: expect an integer

length: the variable that will receive it

&: give scanf the location of that variable

Scanf waits for user input, consumes the characters appropriately from the input typed.

Caution: Output buffering

We may want to show a message/prompt for the user

Keep in mind “output buffering”: The message prompt may not show up always before the scanf, but is stored in a buffer to be printed later.

May result in varying behaviour across systems
Solution: instruct to flush the buffer immediately

```
int length;

printf("Enter length: ");
scanf("%d", &length);

printf("You entered %d\n", length);
```

To make printf print on terminal immediately

Either:

a) Write the following single statement in main():

```
setvbuf(stdout, NULL, _IONBF, 0);
```

OR

b) After every printf that needs immediate display, write:

```
fflush(stdout);
```

Making the cube program interactive

```
#include <stdio.h>

int main(void)
{
    int length, width, height;

    printf("Enter three dimensions: ");
    scanf("%d %d %d",
          &length, &width, &height);

    int volume = length * width * height;
    printf("Volume = %d\n", volume);
    return 0;
}
```

Multiple inputs in a single statement

```
int length, width, height;

scanf("%d %d %d", &length, &width, &height);

/* Input:  4 3 2
   Stores: length = 4
           width  = 3
           height = 2 */
```

Common starter error

```
int length;

/* Correct: scanf receives where length is stored. */
scanf("%d", &length);

/* Incorrect: do not write this. */
scanf("%d", length);
```

The format specifier must match the variable type

C type	scanf	printf	Typical use
int	%d	%d	whole numbers
char	%c	%c	one character
float	%f	%f	about 7 digits precision
double	%lf	%f	about 15–16 digits precision

Important: double uses %lf in scanf, but %f in printf.

Choose a type that matches the kind of value

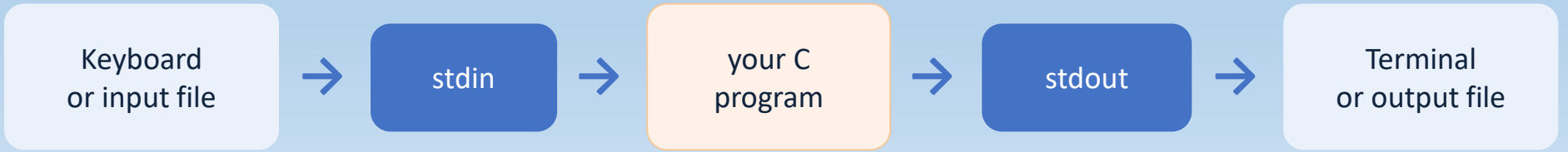
- Use whole-number variables for counts, dimensions, and digit problems.
- Use floating-point variables when a fractional part matters.
- Use a character variable for one symbol, letter, or digit character.
- The variable’s type determines how its stored bits are interpreted.

Area of a circle

```
#include <stdio.h>

int main(void)
{
    double radius, area;

    printf("Enter radius: ");
    scanf("%lf", &radius);

    area = 3.14159 * radius * radius;
    printf("Area = %.2f\n", area);
    return 0;
}
```

Standard Input/Output Streams

- Standard input is the program's default incoming stream.
- Standard output is the program's default outgoing stream.
- In the terminal, the keyboard normally supplies standard input.
- In the terminal, standard output normally appears on the screen.
- Output buffering: things to be printed are stored in stdout, but not printed immediately to the terminal



Caution: character input

```
char grade;
int num;

scanf(“%d”,&num);
printf("Enter a grade letter: ");
scanf(" %c", &grade);

printf("Grade entered: %c\n", grade);

/* A space before %c skips earlier whitespace. */
```

Q: How do we store *strings* (i.e. full words and not just a single character) ? → More in Week 4/5

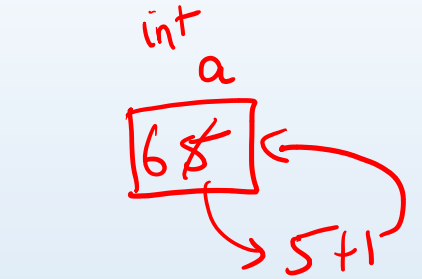
Assignment and Updates

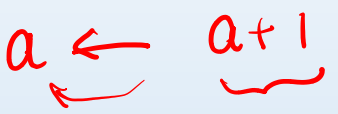
Variables hold a current value that can change

Recap: Declaration, initialization and assignment

```
int score;          /* declaration: create the variable */

score = 10;         /* assignment: store 10 */

int lives = 3;      /* initialization: declare and give
                        an initial value together */
```

Updating a variable



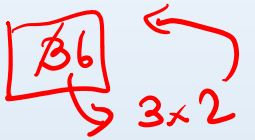



```
int a = 5;
a = a + 1;

/*Will the above compile?*/
```

How is the above done?

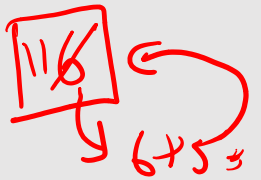
Step 1: read the old value of a -> 5
Step 2: evaluate the right-hand side -> 5 + 1 = 6
Step 3: store the result back in a -> a is now 6 */

What happens at each step




```
int x = 3;

x = x * 2;
x = x + 5;

printf("%d\n", x);
```

value of x = 6



Variable Updates

```
Statement              x before     x after
-----------------------------------------
int x = 3;                –             3
x = x * 2;                3             6
x = x + 5;                6            11

Only the current value remains stored in x.
```

More examples

```
/* Version A */
int a = 10;
a = a + 2;
a = a * 3;
```

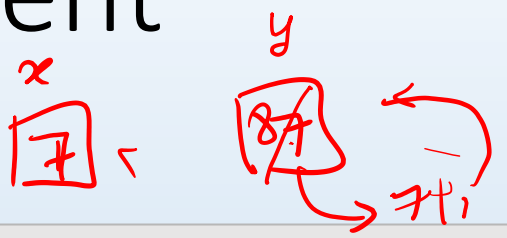
a stores 36

```
 /* Version B */
int b = 10;
b = b * 3;
b = b + 2;
```

b is 32

What is the value of a,b at the end in the above?

More on assignment



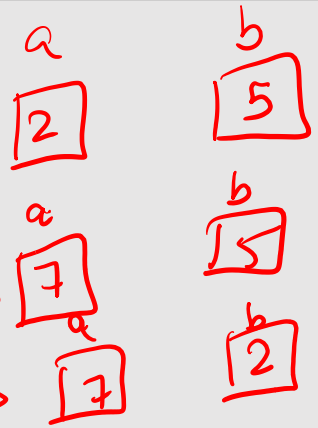
```
int x = 4;
int y = 7;

x = y;          /* copy y's current value into x */
y = y + 1;

printf("%d %d\n", x, y);   /* prints 7 8 */
```

Multiple Updates

```
int balance = 100;

balance = balance + 50;   /* deposit */
printf("After deposit: %d\n", balance);

balance = balance - 30;   /* purchase */
printf("After purchase: %d\n", balance);
```

Predict the final values

```
int a = 2;
int b = 5;

a = a + b;
b = a - b;

printf("a = %d, b = %d\n", a, b);
```



What does the program print?

7 2

Arithmetic Expressions

Precedence order

```
a = b * b + 4;
```

Operations: * / % + -

- Similar to BODMAS rules for math
- Parenthesized expressions are evaluated first.
- Multiplication, division, and **remainder** come next.
- Addition and subtraction come after them.
- Operators at the same level are normally evaluated left to right.

```
int a = 2 + 3 * 4;        /* 14 */
int b = (2 + 3) * 4;      /* 20 */
```