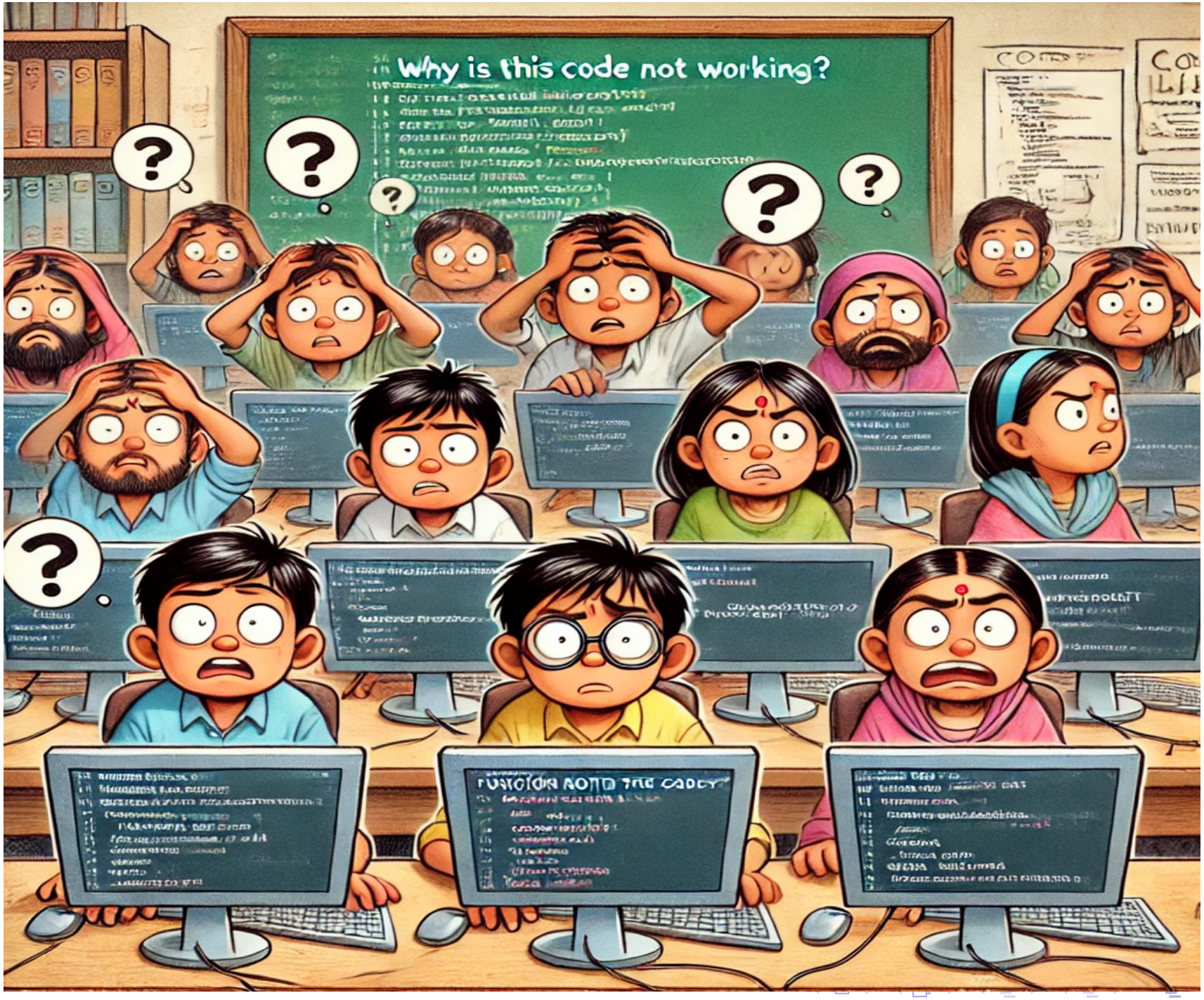
Programming and Data Structures - I

Lecture 3

Kripabandhu Ghosh

CDS, IISER Kolkata

PROGRAMMING ERRORS IN C


Why is this code not working?

Error types

- **Explicit errors**: Lexical, Syntax, Semantic, Linker, Run-time
  - Error message generated
- **Implicit error**: Logical
  - No error message
  - Unexpected output

Explicit errors: Lexical error[1]

Instances

- Token level error – token does not belong to the language
- Illegal characters, spelling error etc.

[1] https://www.javatpoint.com/lexical-error

Explicit errors: Lexical error

Example

```
int a = 20k;
```

Output (compile time)

```
error: fixed-point types not supported for this target
  int i = 20k;
              ^
```

Explicit errors: Syntax error[2]

Instances

- Error in the syntactic structure for the language – parse tree not formed
- Missing symbols (e.g. semicolon), unbalanced parenthesis/bracket etc.

[2] https://www.javatpoint.com/syntax-error

Explicit errors: Syntax error

Example1

```
int a = 20
```

Output (compile time)

```
errors.c:9:1: error: expected ',' or ';' before '}' token
```

Example2

```
int i = 5, b = 6, c;
c = (a + b/2;
```

Output (compile time)

```
errors.c:9:17: error: expected ')' before ';' token
c = (a + b/2;
                ^
```

Explicit errors: Semantic error[3]

Instances

- Mismatch in identifiers of the language – annotated parse tree not formed
- Mismatching arguments, incompatible type of operands etc.

[3] https://www.javatpoint.com/semantic-error

Explicit errors: Semantic error

Example1

```
int a = “kripa”;
```

Output (compile time)

```
errors.c:7:13: warning: initialization makes integer from pointer without a cast
[enabled by default]
int a = “kripa”;
```

Example2

```
int a = 5, b;
char c[] = “kripa”;
b = a - c;
```

Output (compile time)

```
errors.c:10:11: error: invalid operands to binary – (have ‘int’ and ‘char *’)
b = a - c;
        ^
```

Explicit errors: Linker error[4]

Instances

- Linker not able to generate the executable code – failure to link with dependent modules
- Use of wrong function prototype, wrong header file etc.

[4] https://www.javatpoint.com/programming-errors-in-c

Explicit errors: Linker error

Example

Printf(“Kripa”);

Output (compile time)

errors.c:(.text+0x1a): undefined reference to ‘Printf’
collect2: error: ld returned 1 exit status

Explicit errors: Runtime error[5]

Instances

Error during execution of the program after successful compilation

[5] https://www.javatpoint.com/programming-errors-in-c

Explicit errors: Runtime error

Example

```
int i = 5, j = 0, k;
k = i/j;
```

Output (execution time)

```
Floating point exception (core dumped)
```

Implicit error: logical error[6]

Instances

- Error in programming logic
- No compilation or execution error
- Desired output not obtained

[6] https://www.javatpoint.com/programming-errors-in-c

Implicit error: logical error (Example)

Program

```
#include<stdio.h>
void main()
{
   int sum = 0, i;
   for(i = 1; i <= 10; i++); // logical error, the loop body not executed
   {
      sum = sum + 1;
   }
   printf("The value of sum is %d\n", sum);
}
```

Output (execution time)

The value of sum is 1

CONSTANT, VARIABLES AND DATA TYPES

Character Set

Character category	**Examples**
Letter	Uppercase (A-Z) Lowercase (a-z)
Digits	0-9
Special characters	comma (,), period (.), semicolon (;)
White spaces	Back space, horizontal tab, New line

Tokens

Type	Examples
Keywords	float, while
Identifiers	main
Constants	-15.5, 50
Strings	“ABC”, “xyz”
Special Symbols	, []
Operators	+, -, *

Constants

Type	Examples
Numeric	Integer Real
Character	Single character String

Integer Constants

- No decimal point
- Positive or Negative (if no sign preceeds, assumed to be positive)
- No commas or blanks are allowed within an integer constant
- Range of values is compiler dependent
  - The allowable range for integer constants for 16-bit compiler is -32768 to 32767 (-2^{15} to $+2^{15}-1$)
  - One bit for sign and rest for magnitude

Integer Constants

Type	Examples
Decimal	123, -123, 0, +55
Octal (0–7)	037, 0553
Hexadecimal (0–9, A–F)	0x, 0x3, 0x9F

Decimal, Octal and Hexadecimal

Program

```
#include<stdio.h>

int main()
{
   int dec = 11;
   int oct=0111;
   int hex = 0xAA;

   printf("\nThe decimal equivalents:\n");
   printf("%d\n", dec);
   printf("%d\n", oct);
   printf("%d\n", hex);

   printf("\nThe decimal, octal and hexadecimal:\n");
   printf("%d\n", dec);
   printf("%o\n", oct);
   printf("%x\n", hex);
   return 0;
}
```

Decimal, Octal and Hexadecimal (contd.)

Output

The decimal equivalents:
11
73
170

The decimal, octal and hexadecimal:
11
111
aa

Decimal to Decimal

Base 10

$$11_{10} = 1 \times 10^1 + 1 \times 10^0 = 10 + 1 = 11_{10}$$

Octal to Decimal

Base 8

$$111_8 = 1 \times 8^2 + 1 \times 8^1 + 1 \times 8^0 = 64 + 8 + 1 = 73_{10}$$

Hexadecimal to Decimal

Base 16

$AA_{16} = 10 \times 16^1 + 10 \times 16^0 = 160 + 10 = 170_{10}$

Real Constants

- Has decimal point
- Positive or Negative (if no sign preceeds, assumed to be positive)
- No commas or blanks are allowed within an integer constant
- Range of values is compiler dependent
  - The allowable range for integer constants for 16/32-bit compiler is 3.4E-38 to 3.4E+38
  - One bit for sign and rest for exponent (e.g. 8 bits) and fraction (e.g. 23 bits) (IEEE Standard 754)

Real Constants

Type	Examples
Floating point	0.012, -0.66, 123.45, +23.67, 230., .95, -.78, +.45
Exponential (*mantissa* e/E *exponent*) *mantissa* : floating point/decimal *exponent* : decimal	7.5E9 (7500000000)

Numeric Constants: Summary

Example	Validity	Remark
25,000	✗	Comma is not allowed
75e 4	✗	White space is not allowed
1.5E2.5	✗	Exponent must be an integer
$123	✗	Non-digit character not allowed
12345L	✓	Long integer
0x39039ABCDEF	✓	Hexadecimal long integer

String Constants

Type	Examples
Single characters	'5', 'a', ' '
Backslash characters	\n (new line) \t (horizontal tab) \0 (null)
Strings	"I am a string" "5" (different from '5')

String Constants

Type	Examples
Single characters	'5', 'a', ' '
Backslash characters	\n (new line) \t (horizontal tab) \0 (null)
Strings	"I am a string" "5" (different from '5')

Single quotes are different from double quotes in C!!

Single character and String

ASCII value

```
printf("ASCII code of a is %d", 'a');
printf("My name is SRK!");
```

Output

```
ASCII code of a is 97
My name is SRK!
```

Variable

- Data name used to store a constant value
- Usually can be updated
- Memory allocation done for a variable

Variable address

```
int a = 5;
int *b;
b = &a;
```

Variable name rules

- Must start with a letter (some compilers allow variables starting with an underscore)
- Usually should not exceed 31 characters in length; some compilers recognize only the first 8 characters (unnecessarily long variable names should be avoided)
- Case sensitive (Sum, sum and SUM are different)
- Should not be a keyword (auto, break, printf etc.)
- White space not allowed

Data types

- Basic (fundamental) data types
- User-defined data types
- Derived datatypes (arrays, structures etc.)

Basic data types

- Integral (int)
- Character (char)
- Floating point (float, double, long double)

Datatypes for a 16-bit machine[7]

Data Type	Range	Bytes	Format
signed char	-128 to + 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf
Note: The sizes and ranges of int, short and long are compiler dependent. Sizes in this figure are for 16-bit compiler.			

[7] Table reused from *Let Us C*. Yashavant P. Kanetkar. 5th Ed.

Printing data type size[8] and range

Code

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

void main()
{
   printf("sizeof(char) = %u\n",sizeof(char));
   printf("sizeof(short) =%u\n",sizeof(short));
   printf("sizeof(int) = %u\n",sizeof(int));
   printf("sizeof(long) = %u\n",sizeof(long));
   printf("sizeof(float) = %u\n",sizeof(float));
   printf("sizeof(double) = %u\n",sizeof(double));
   printf("sizeof(long double) = %u\n",sizeof(long double));

   printf("SCHAR_MIN = %d\n",SCHAR_MIN);
   printf("SCHAR_MAX = %d\n",SCHAR_MAX);
   printf("UCHAR_MAX = %d\n",UCHAR_MAX);
```

[8] **sizeof** generates the storage size of an expression or a data type

Printing data type size and range (contd.)

Code

```
    printf("SHRT_MIN = %d\n",SHRT_MIN);
    printf("SHRT_MAX = %d\n",SHRT_MAX);
    printf("USHRT_MAX = %u\n",USHRT_MAX);
    printf("INT_MIN = %d\n",INT_MIN);
    printf("INT_MAX = %d\n",INT_MAX);
    printf("UINT_MAX = %u\n",UINT_MAX);
    printf("LONG_MIN =%ld\n",LONG_MIN);
    printf("LONG_MAX = %ld\n",LONG_MAX);
    printf("ULONG_MAX = %lu\n",ULONG_MAX);
    printf("FLT_MIN = %e\n",FLT_MIN);
    printf("FLT_MAX = %e\n",FLT_MAX);
    printf("DBL_MIN = %e\n",DBL_MIN);
    printf("DBL_MAX = %e\n",DBL_MAX);
    printf("LDBL_MIN = %e\n",LDBL_MIN);
    printf("LDBL_MAX = %e\n",LDBL_MAX);
}
```

Printing data type size and range (contd.)

Output

```
sizeof(char) = 1
sizeof(short) = 2
sizeof(int) = 4
sizeof(long) = 8
sizeof(float) = 4
sizeof(double) = 8
sizeof(long double) = 16
SCHAR_MIN = -128
SCHAR_MAX = 127
UCHAR_MAX = 255
SHRT_MIN = -32768
SHRT_MAX = 32767
USHRT_MAX = 65535
```

Printing data type size and range (contd.)

Output (contd.)

INT_MIN = -2147483648
INT_MAX = 2147483647
UINT_MAX = 4294967295
LONG_MIN = -9223372036854775808
LONG_MAX = 9223372036854775807
ULONG_MAX = 18446744073709551615
FLT_MIN = 1.175494e-38
FLT_MAX = 3.402823e+38
DBL_MIN = 2.225074e-308
DBL_MAX = 1.797693e+308
LDBL_MIN = 1.797693e+308
LDBL_MAX = 1.797693e+308

Storage Classes

Name	Meaning
auto (default)	Local variable known only to the function in which it is declared
static	Local variable which exists and retains its value even after control is returned to the calling function
extern	Global variable known to all functions
register	Local variable stored in a register

- Global (extern) and static variables are automatically initialized to zero
- Local variables have garbage values unless explicitly initialized

Thanks