INTRODUCTION TO C

Operating Systems

Presenter: Ali Momen

TABLE OF CONTENTS

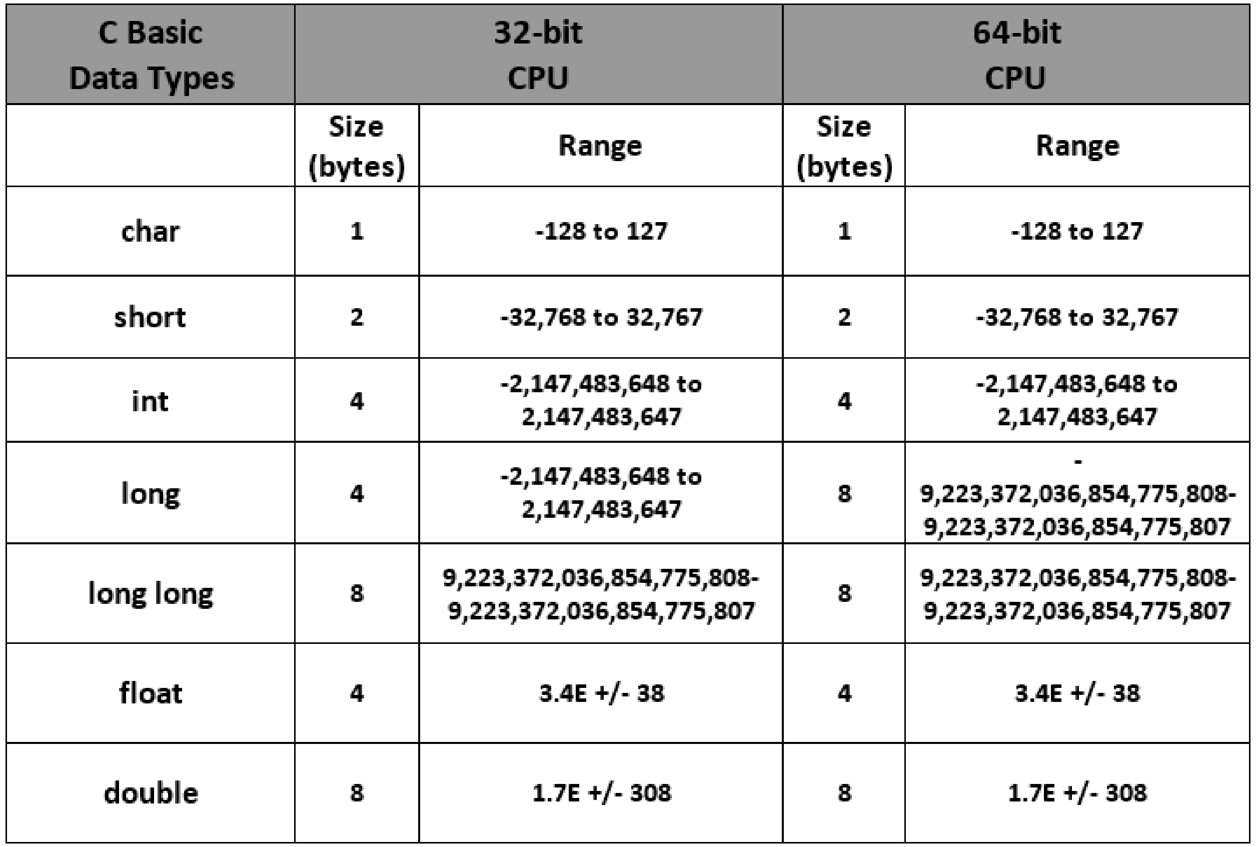
STRUCTURE OF A .C FILE

```
/* Begin with comments about file contents */

Insert #include statements and preprocessor
definitions

Function prototypes and variable declarations

Define main() function
{
  Function body
}

Define other function
{
  Function body
}
⋮
```

VARIABLE DECLARATION AND INITIALIZATION

- Must declare variables before use
- Variable declaration & initialization:

```
int n;
float phi = 1.678;
```



C Basic Data Types	32-bit CPU		64-bit CPU	
	Size (bytes)	Range	Size (bytes)	Range
char	1	-128 to 127	1	-128 to 127
short	2	-32,768 to 32,767	2	-32,768 to 32,767
int	4	-2,147,483,648 to 2,147,483,647	4	-2,147,483,648 to 2,147,483,647
long	4	-2,147,483,648 to 2,147,483,647	8	- 9,223,372,036,854,775,808- 9,223,372,036,854,775,807
long long	8	9,223,372,036,854,775,808- 9,223,372,036,854,775,807	8	9,223,372,036,854,775,808- 9,223,372,036,854,775,807
float	4	3.4E +/- 38	4	3.4E +/- 38
double	8	1.7E +/- 308	8	1.7E +/- 308

OPERATORS IN C

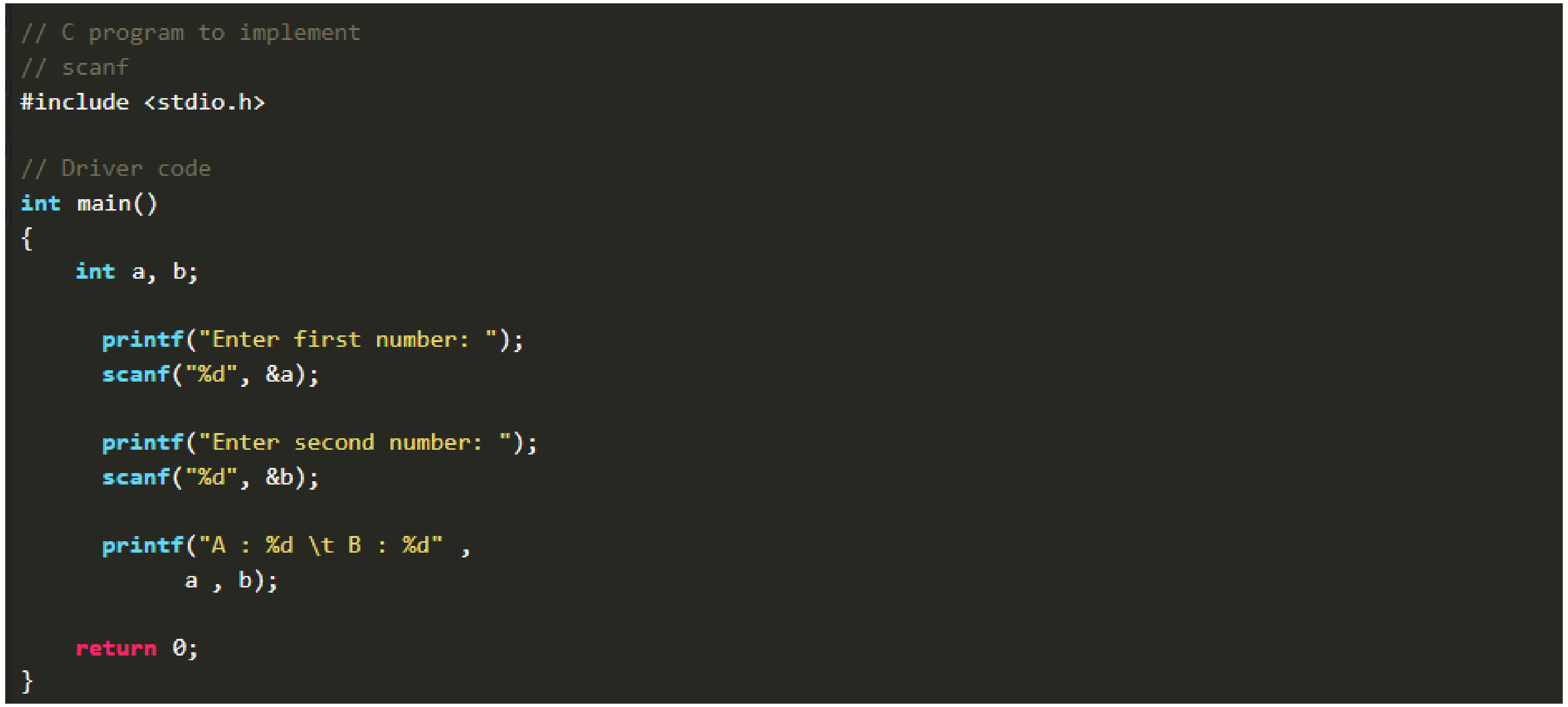
- Operators are symbols that perform operations on variables and values.
- Common operators in C include arithmetic operators (+, -, *, /), relational operators (==, !=, <, >), and logical operators (&&, ||, !).
- Understanding operator precedence is crucial for writing correct expressions.

I/O IN C

- We use scanf and printf for I/O
- Scanf gets a pointer to the variable but printf uses the value

- *a ---- &a

- %d %ld %lld %c %s %p



```
// C program to implement
// scanf
#include <stdio.h>

// Driver code
int main()
{
    int a, b;

      printf("Enter first number: ");
      scanf("%d", &a);

      printf("Enter second number: ");
      scanf("%d", &b);

      printf("A : %d \t B : %d" ,
            a , b);

    return 0;
}
```

STRINGS IN C

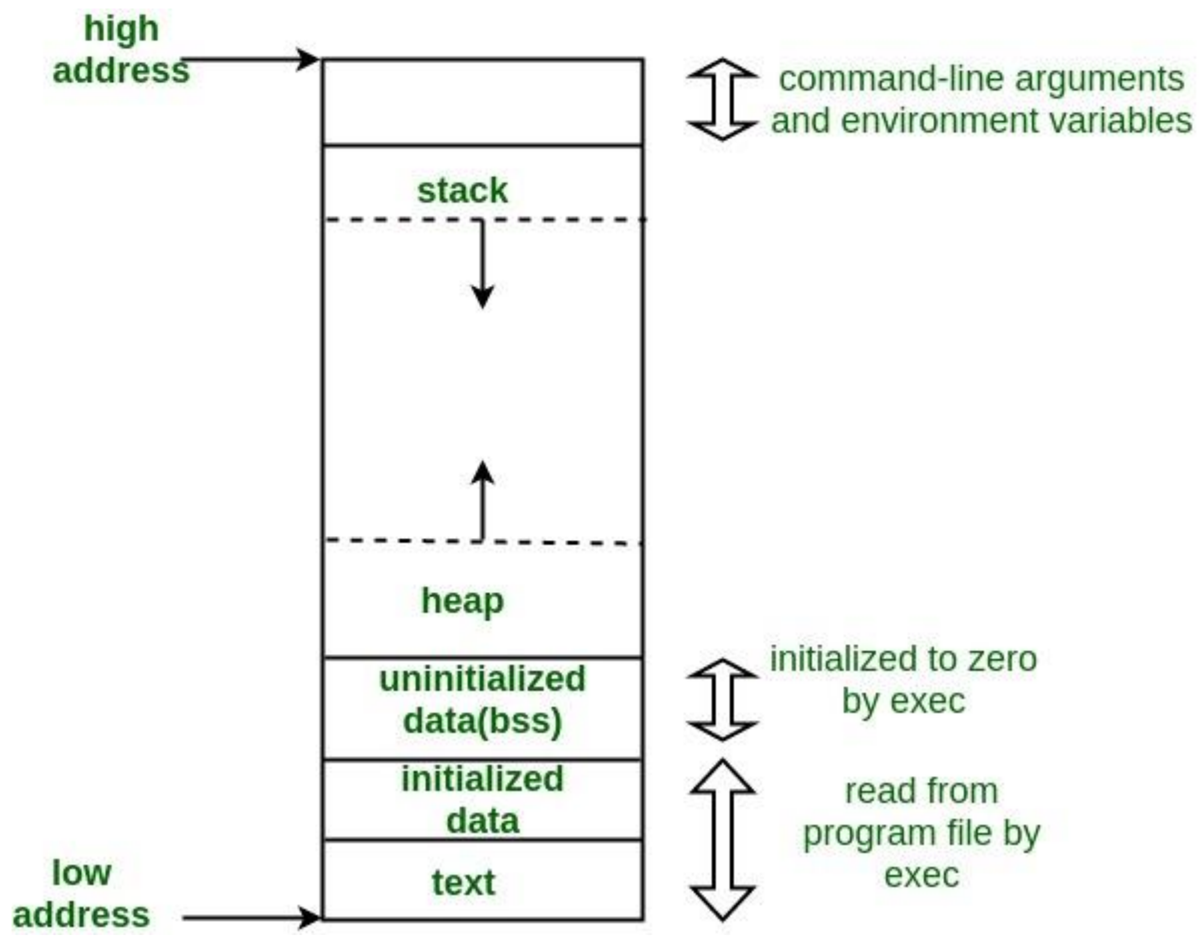
```
#include <stdio.h>
#define MAX_LIMIT 20
int main()
{
   char str[MAX_LIMIT];
   fgets(str, MAX_LIMIT, stdin);
   printf("%s", str);

   return 0;
}
```

MEMORY IN C

*Stack
LIFO structure- in intel processors it grows downwards
What is a stack frame??
The reason we cannot access local variables from other functions.

- Heap
- Dynamic data structure – you have to use malloc to store your data on heap

- Bss for global or static data (uninitialized)
- Text = your code


high address
command-line arguments and environment variables
stack
heap
uninitialized data(bss)
initialized to zero by exec
initialized data
read from program file by exec
text
low address

INTEL REGISTERS

Examples include EAX, EBX, ECX, and EDX.

EAX → for storing data for various operations
EBX → base pointer – is pushed to the stack from function to function
ECX → Loop counting
EIP → Instruction pointer
ESP → Stack Pointer

UNDERSTANDING POINTERS

- Pointers in C are variables that store memory addresses.

- To declare a pointer in C, you specify the data type it points to, followed by an asterisk (*), and the pointer name.

- Int a = 6;
- Int* x = &a;
- The dereferencing operator (*) is used to access the value stored at a particular memory address.
- X or *x or &x

MALLOC AND HEAP

- Dynamic memory allocation in C allows for allocating memory at runtime.

- Syntax: ptr = (cast-type*) malloc(byte-size);
- Example: int* ptr = (int*) malloc(5 * sizeof(int));
- int** Row = (int**) malloc (5 * sizeof(int*));
- for(int i=0; i<5;i++){
- Row[i]= (int*) malloc(10 * sizeof(int));
- }

GDB / ASAN / TSAN

- Segmentation fault??
- break linenumber – create breakpoint at specified line
- run – run program
- c – continue execution
- next – execute next line
- step – execute next line or step into function
- quit – quit gdb
- print expression – print current value of the specified expression
- help command – in-program help

THANK YOU