Basics of C

Data and File Structures Laboratory

http://www.isical.ac.in/~dfslab/2017/index.html

Agenda

- In-built types and operators
- Conditional statements (`if`): executing statements based on whether some condition holds or does not hold
- Loops: executing statements repeatedly
- Input and output

Variable names

- REQUIRED
  - must start with a letter or underscore (_)
  - can contain only letters, underscores, digits
  - cannot match *reserved* words
  - case-sensitive

Variable names

- REQUIRED
  - must start with a letter or underscore (_)
  - can contain only letters, underscores, digits
  - cannot match *reserved* words
  - case-sensitive
- RECOMMENDED
  - use “meaningful” names
  - use `under_scores` or `CamelCase` for long names
  - Hungarian notation (much debated!)
    - `https://en.wikipedia.org/wiki/Hungarian_notation`
    - `http://stackoverflow.com/questions/111933/why-shouldnt-i-use-hungarian-notation`
    - `http://stackoverflow.com/questions/202107/good-examples-of-hungarian-notation`

In-built types

Integer data types

Type	Size**	Minimum value	Maximum value
char	8	-2^7	$2^7 - 1$
short int	16	-2^{15}	$2^{15} - 1$
int	32	-2^{31}	$2^{31} - 1$
long int	32	-2^{31}	$2^{31} - 1$
long long int	64	-2^{63}	$2^{63} - 1$
unsigned char	8	0	$2^8 - 1$
unsigned short int	16	0	$2^{16} - 1$
unsigned int	32	0	$2^{32} - 1$
unsigned long int	32	0	$2^{32} - 1$
unsigned long long int	64	0	$2^{64} - 1$

In-built types

“Real” (floating point) numbers

Type	Size
float	32
double	64
long double	128

Examples:

```
1.23456     3.45e67
1.          +3.45e67
.1          -3.45e-67
-0.12345    .00345e-32
+.4560      1e-15
```

In-built types

“Real” (floating point) numbers

Type	Size
float	32
double	64
long double	128

Examples:

```
1.23456     3.45e67
1.          +3.45e67
.1          -3.45e-67
-0.12345    .00345e-32
+.4560      1e-15
```

- Do not use commas as thousand-separators.
- At times behaviour may be counter-intuitive (more about this later).

Operators

Arithmetic operators: `+ - * / %`

Increment decrement operator: `++ --`

Relational operators: `< <= == != >= >`

Operators

Arithmetic operators: `+ - * / %`

Increment decrement operator: `++ --`

Relational operators: `< <= == != >= >`

Boolean operators: `&& || !`

Boolean values

Any non-zero value is **TRUE**; zero is **FALSE**

Examples:

0	False	0e10	False
1	True	'A'	True
6 - 2 * 3	False	'\0'	False
(6 - 2) * 3	True	x = 0	False
0.0075	True	x = 1	True

Conditionals: if, if-else

```
if (condition) {
    statements
}
```

```
if (condition) {
    statements
}
else {
    statements
}
```

Conditionals: switch

```
switch (E) {
    case value1 :
       statement;
       break;
    case val2 :
       statement;
       break;
    ...
    case valn :
       statement;
       break;
    default:
       statement;
 }
```

Loops

```
while (condition) {
        statement;
}
```

Loops

```
while (condition) {
      statement;
}
```

```
do {
     statement;
} while (condition);
```

Loops

```
while (condition) {
      statement;
}
```

```
do {
     statement;
} while (condition);
```

```
for ( initialisation ; condition ; update operation ) {
    statement;
}
```

Loops

- **break:** immediately jump to the next operation after the loop
- **continue:** do the update operation if applicable, and continue with the next iteration of the loop

Loops

- **break:** immediately jump to the next operation after the loop
- **continue:** do the update operation if applicable, and continue with the next iteration of the loop

```
for (i=1; i<=100; ++i) {
     printf("%4d",i);
     if (i%10 != 0)
        continue;
     printf("\n");
}
```

Loops

- **break:** immediately jump to the next operation after the loop
- **continue:** do the update operation if applicable, and continue with the next iteration of the loop

```
for (i=1; i<=100; ++i) {
     printf("%4d",i);
     if (i%10 != 0)
        continue;
     printf("\n");
}
```

```
i = 0;
while (i < 100) {
    ++i;
    printf("%4d",i);
    if (i%10 != 0) continue;
    printf("\n");
}
```

Input / output

I/O from the terminal

- **printf**, **scanf**
- **getchar**, **putchar**

Input / output

I/O from the terminal

- **printf**, **scanf**
- **getchar**, **putchar**

I/O from files

- File pointers: `FILE *`
- **fopen**, **fclose**
- **fprintf**, **fscanf**
- **fgetc**, **fputc**
- **fgets**, **fputs**

Input / output

I/O from the terminal

- **printf**, **scanf**
- **getchar**, **putchar**

I/O from files

- File pointers: `FILE *`
- **fopen**, **fclose**
- **fprintf**, **fscanf**
- **fgetc**, **fputc**
- **fgets**, **fputs**

- Practice reading man pages.
  e.g. `$ man fopen`
- **Do not use** `gets()`**!**

Acknowledgements

- `http://cse.iitkgp.ac.in/~pds/notes/`
  (please see the above page for many more practice problems)