CSE 1201: Fundamentals of Programming

Lecture 2

Computer was modeled after Human

Sensory Memory
(<1 sec)

Short Term Memory
(<1 minute)

Long Term Memory
(Life Time)

http://www.human-memory.net/types_sensory.html

A simple model of the computer

Definition: Computer Program

A Computer program is a sequence of instructions written to perform a specified task with a computer.

Example: Computer Program

Computer tell me what would be my balance after 1 year if my starting balance is 1,00,000 and interest rate is 10%.

Introduction to C

- C programming language
  - Structured and disciplined approach to program design

Story	Paragraph	Sentence
Program	Function	Statement

A Simple C Code

```
#include <stdio.h>

int main()
{
 printf(“This is CSE 1201”);
 return 0;
}
```

How to Run

- Save a source code with extension “c” or “cpp”. (Ex: **first.c**)
- Compile it and link it
    - Output: first.exe
    - Run the program.
- Output of the program:
    - `This is CSE 1201`

A Simple C Code : Printing a line

```
1        /* Fig. 2.1: fig02_01.c
2      A first program in C */
3 #include <stdio.h>
4
5 int main()
6  {
7  printf( "Welcome
   to CSE 1201!\n" );
8      return 0;
9
10 }
```

```
Welcome to CSE 1201!
```

- Comments
  - Text surrounded by **/*** and ***/** is ignored by computer
  - Used to describe program
- **#include <stdio.h>**
  - Preprocessor directive - tells computer to load contents of a certain file
  - **<stdio.h>** allows standard input/output operations

A Simple C Code : Printing a line (II)

- **int main()**
  - C programs contain one or more functions, exactly one of which must be **main**
  - Parenthesis used to indicate a function
  - **int** means that main "returns" an integer value
  - Braces indicate a block
    - The bodies of all functions must be contained in braces

A Simple C Code : Printing a line (III)

- **printf("Welcome to C!\n");**
  - Instructs computer to perform an action
    - Specifically, prints string of characters within quotes
  - Entire line called a statement
    - All statements must end with a semicolon
  - ****- escape character
    - Indicates that **printf** should do something out of the ordinary
    - **\n** is the newline character

A Simple C Code : Printing a line (IV)

- **return 0;**
  - A way to exit a function
  - **return 0**, in this case, means that the program terminated normally
- Right brace **}**
  - Indicates end of **main** has been reached

Basic Data Types

Type	Example	Size	Range	Format Code
char	A B C … Z a b c … z @ $ % *	8 bits, 1 byte	*-128 to +127*	"%c"
int	32, 3, 4 ,5 -4, -5, -6	32 bits, 4 bytes	*-2,14,74,83,648 to +2,14,74,83,647*	"%d" "%i"
float	3.4, 5.25, -2.3, -6.7	32 bits, 4 bytes	3.4×10^{-38} to 3.4×10^{38}	"%f"
double	3.4, 5.25, -2.3, -6.7	64 bits, 8 bytes	1.7×10^{-308} to $1.7\times10^{+308}$	"%lf"

Float Format

Single precision floating point:

- Sign bit: 1
- Exponent: 8 bits
- Mantissa: 23 bits
- It can store decimal values with precision up to 6-7 decimal places.

Double Format

Double precision floating point:

Sign bit: 1

Exponent: 11 bits

Mantissa: 52 bits

It can store values up to **15 decimal points** without loss of precision.

ASCII Values

TABLE 3
ASCII CHARACTER CODES (DECIMAL)

0	Ctrl-@	32	Space	64	@	96	‘
1	Ctrl-A	33	!	65	A	97	a
2	Ctrl-B	34	"	66	B	98	b
3	Ctrl-C	35	#	67	C	99	c
4	Ctrl-D	36	$	68	D	100	d
5	Ctrl-E	37	%	69	E	101	e
6	Ctrl-F	38	&	70	F	102	f
7	Ctrl-G	39	’	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Tab	41	)	73	I	105	i
10	Ctrl-J	42	*	74	J	106	j
11	Ctrl-K	43	+	75	K	107	k
12	Ctrl-L	44	,	76	L	108	l
13	Return	45	-	77	M	109	m
14	Ctrl-N	46	.	78	N	110	n
15	Ctrl-O	47	/	79	O	111	o
16	Ctrl-P	48	0	80	P	112	p
17	Ctrl-Q	49	1	81	Q	113	q
18	Ctrl-R	50	2	82	R	114	r
19	Ctrl-S	51	3	83	S	115	s
20	Ctrl-T	52	4	84	T	116	t
21	Ctrl-U	53	5	85	U	117	u
22	Ctrl-V	54	6	86	V	118	v
23	Ctrl-W	55	7	87	W	119	w
24	Ctrl-X	56	8	88	X	120	x
25	Ctrl-Y	57	9	89	Y	121	y
26	Ctrl-Z	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	Ctrl-\	60	<	92	\	124	|
29	Ctrl-]	61	=	93	]	125	}
30	Ctrl-^	62	>	94	^	126	~
31	Ctrl-_	63	?	95	_	127	Delete

Memory Concepts

- **Variables**
  - Variable names correspond to locations in the computer's memory.
  - Every variable has a name, a type, a size and a value.
  - Whenever a new value is placed into a variable (through **scanf**, for example), it replaces (and destroys) previous value
  - Reading **variables** from memory does not change them

- A visual representation

Using Variables - Output

```
int a, b, c;

a = 3;

b = 8;

c = a + b;

printf("%d" , c);
```

Output:

```
11
```

Using Variables – Output (II)

```
int a, b, c;

a = 3;

b = 8;

c = a + b;

printf("The value of c:");

printf("%d" , c);
```

Output:

```
The value of c : 11
```

Using Variables – Output (II)

```
int a, b, c;

a = 3;

b = 8;

c = a + b;

printf("The value of c: %d",c);
```

Output:

```
The value of c : 11
```

Using Format Specifier

Format specifier

Variable

```
printf(type, what);
```

Type	Format Code
char	"%c"
int	"%d" "%i"
float	"%f"
double	"%lf"

Using Variables - Output

```
int a, b, c;

a = 3;

b = 8;

c = a + b;

printf(“%d” , c);
```

```
char a = 90;

printf("%c", a);
```

```
float a = 8.958;

printf("%f ", a);
```

```
double a = -9.8;

printf("%lf ", a);
```

```
char a = ‘Z’;

printf("%c", a);
```

```
char a = ‘Z’;

printf("%d", a);
```

Data Input

```
int a;
scanf("%d",&a);
```

```
char c;
scanf("%c",&c);
```

```
float b;
scanf("%f",&b);
```

```
double d;
scanf("%lf",&d);
```

Quiz: What is the output if user enters 0

```
int a;
scanf("%d",&a);
printf("%d",a);
```

```
char c;
scanf("%c",&c);
printf("%d",c);
```

Naming convention of a variable

- Capital letters A-Z, lowercase letters a-z, digits 0- 9, and the underscore character
- First character must be a letter or underscore
- Usually only the first 31 characters are significant
- There can be no embedded blanks
- Keywords cannot be used as identifiers
- Identifiers are case sensitive

Keywords in C

Keywords			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Some valid/invalid variable names

- OK First_tag
- NO char
- NO Price$
- NO group one
- OK average_number
- OK int_
- NO 8boys

Declaring Variables

```
1    /* Using variable in C
*/
2 #include <stdio.h>
3
4
5    int main()
6    {
7     int interestrate;
8     char ch;
9     return 0;
10   }
```

Assigning Values

- Two ways:
  - Using **scanf**

```
int c;
scanf("%d",&c);
```

  - Using **assignment operator**

```
int c;  c = 10;
```

Arithmetic Operators on int & char

- Arithmetic operators:

C Operation	Arithmetic operator	Algebraic Expression	C Expression
Addition	+	*f + 7*	f + 7
Subtraction	-	*p – c*	p - c
Multiplication	*	*bm*	b * m
Division	/	*x / y*	x / y
Modulus	%	*r mod s*	r % s

- Rules of operator precedence:

Operator(s)	Operation(s)	Order of evaluation (precedence)
()	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
*, /, or %	Multiplication Division Modulus	Evaluated second. If there are several, they are evaluated left to right.
+ or -	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Arithmetic (II)

- Arithmetic calculations are used in most programs
  - Use ***** for multiplication and **/** for division
  - Integer division truncates remainder
    7/5 evaluates to **1**
  - Modulus operator returns the remainder
    7%5 evaluates to **2**
- Operator precedence
  - Some arithmetic operators act before others (i.e., multiplication before addition)
    - Use parenthesis when needed
  - Example: Find the average of three variables **a**, **b** and **c**
    - Do not use: **a + b + c / 3**
    - Use: **(a + b + c) / 3**

Operator Precedence Example

- Find the values of the followings:

5+6*7-8/7	=	46
5*4/3	=	6
5/(6+7) – 8%8	=	0

Working with Characters

- Take a character as input and print ASCII value.
- Convert uppercase to lower case and vice versa
- Write down a program that will take a capital letter and an integer N as input and will print n-th letter starting from the letter given as input (wrap around). Assume that n is a nonnegative integer less than or equal to 25. (You can not perform condition checking to solve this problem).

Solution to 3

```
int main( ){
    int npos,cpos,n;
    char ch,nch;
    scanf("%c %d",&ch,&n);
    cpos = ch - ‘A';
    npos = (cpos +n)%26;
    nch = npos + ‘A';

    printf("%c",nch);

    return 0;

}
```

Working with int (Examples)

- To extract digits of a number:
  - Given a 4 digit number, can you reverse it?
  - Modulus (%) and division (/) operators are good enough.

- Interchange content of two variables:
  - ❑ Using one additional variable
  - ❑ Using no additional variable

- Time difference of two cities.
  - Dhaka: 11:20
  - Mumbai: 10:50

Compound Operators

When one is only interested in updating the value of a variable it is possible to use compound assignment.

a += b; (-= , *= , /=)

is equivalent to

a = a + b; (-,*,/).

i=i+1 can be conveniently written using the increment operator ++.
i=i+1 can be written i++ (postfix) and ++i (prefix).

Prefix: i = 1; j = 1; k = j + ++i;
i ← 2,k ← 3 (=1+2)

i updated before evaluation

Postfix: i = 1; j = 1; k = j + i++;
k ← 2 (=1+1) , i ← 2
i updated after evaluation

Decrement operator : --

Formatted Output

```
printf("Area = %f\n", pi*r*r);
```

"Area = %f\n" is a format string instructing the computer to write the string "Area = ” followed by a floating point number (pi*r*r evaluated) and a linebreak to stdout.\

The format string must contain a conversion specification for each value to print.

Some common conversion specifiers:

Specifier	**Displays**	**Functionality**
%f	Floating point numbers	represents the data in normal decimal form, upto six decimal places
%e	exponential form	represents the data in exponential power(scientific format).
%g	Combination of two above	Represents %f (a simple float or double) or %e (scientific notation) and returns it as the shorter of the two.

Formatted Output & Escape Sequences

The basic specifiers can be modified. The specification %p.qX prints a value of type X with precision q and a minimum field width p
(for tables).

The meaning of “precision” depends on X. Typically the number of decimals (X = f or X = e) or significant digits (X = g or X = d).

Sequence	**Displays**
\t	Tab
\"	Double quote (”)
\\	Backslash (\)
\?	Question mark (?)

References

- https://www.geeksforgeeks.org/c-float-and-double/
- https://stackoverflow.com/questions/54162152/what-precisely-does-the-g-printf-specifier-mean

Any Questions?

Thank You