GNG1106
Fundamentals of Engineering Computation

Instructor: **Hitham Jleed**

University of Ottawa

Fall 2023 ~

In-Class Exercise:

Outline

1. The `char` Type
2. More on Decision Structures

Outline

1 The char Type

2 More on Decision Structures

- In C, a character, such as ‘a’, ‘x’, ‘?’, ‘$’, ... , is stored in a variable of `char` type.
- A `char` typed value takes one byte (i.e., 8 bits) to store.
- A character is represented (or “encoded”) as an integer (which is represented in binary) inside computers.
- The most common method for encoding characters into integers (or binary strings) is the ASCII (American Standard Code for Information Interchange) Code.
- The ASCII code represents 128 characters and special symbols using 7 bits.
- Under the ASCII encoding, each character has an integer value and a 7-bit representation, which fits in the one-byte storage for the `char` type.

ASCII Table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Picture taken from `https://simple.wikipedia.org/wiki/ASCII`

- The characters in the range 1 through 31 inclusively, are special characters.
- For example, the character [BELL] (numerical value 7) can be used to ring the bell on the computer.
- The ASCII codes for characters '0' through '9' do not have numerical value 0 to 9
- Note that characters 'A' through 'Z' and 'a' through ''z have ASCII codes in increasing order of their numerical values.
  - Lowercase letter = uppercase letter + 32
- In C, the notation for a character literal requires single quotation marks, such as, `'a'`, `'B'`, `'3'`, `'&'`, `'$'`, `'\n'` ...

- Declaring a `char`-typed variable: `char myChar;`
- Assigning values to a `char`-typed variable: Two ways.
  - `myChar = 'a'; // note: single quotes, not double quotes!`
  - `myChar = 97;`
- Declaration with initialization:
  - `char myChar='a';`
  - `char myChar=97;`
- A variable of type char can be interpreted either as a character or as an integer.
  - `printf("%d", myChar) //print as an integer`
  - `printf("%c", myChar) //print as a character`

- As a number, a char-typed value has range from -128 to 127.
- We can also use int-typed variables to store characters, but caution that as a number, a `char` has smaller range than an `int`
  - `int A=353; printf("%c", A);`
  - On MacOS, this appears to print `'a'` (since 353%256=97=`'a'`)
- Since char-typed values are merely numbers (which can be interpreted as characters), arithmetic, equality, and relational operations can all be applied to them (except that their range is rather small) as they do to other fixed-point values.

expression	value
`'a'+'b'`	195 (=97+98)
`'a' == 'b'`	0 (FALSE)
`'a'<'b'`	1 (TRUE)

Read Characters from the Keyboard

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

Note

Caution! scanf() is known to be fragile for reading characters.

The Problem of `scanf` in Reading Characters

```
char myChar1, myChar2;
scanf("%c", &myChar1);
printf("myChar1 is %c\n",
myChar1);
scanf("%c", &myChar2);
printf("myChar2 is %c\n",
myChar2);
```

- When executing this code, if you hit the ENTER key after entering the first character for the first call of `scanf`, the code also automatically executes the second `scanf` and the ENTER key (the character `'\n'`) is read and loaded into variable `myChar2`.
- It is recommended that a line "`fflush(stdin);`" be added before every call of `scanf`. But depending on the OS and compiler, this may or may not solve the issue.
- Later we will introduce better ways to read characters.

Coding Demonstration

Outline

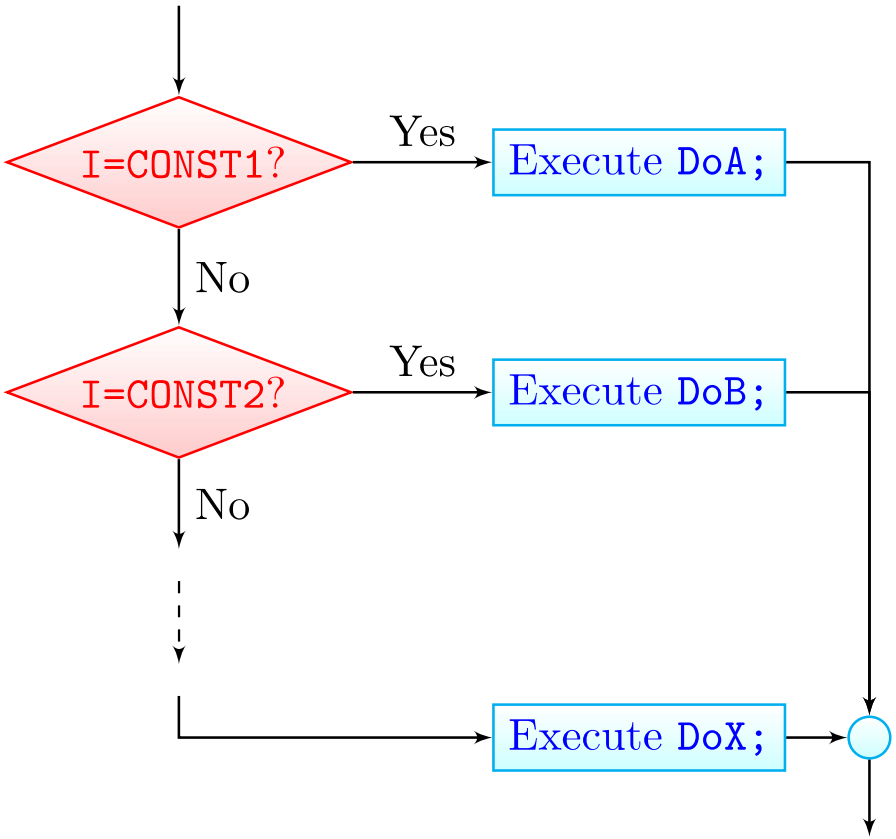
1 The char Type

2 More on Decision Structures

switch Statement

```
switch (I)
{
   case CONST1:
      DoA;
      break;
   case CONST2:
      DoB;
      break;
   ...
   default:
      DoX;
}
```

- switch, case, default, and break are all keywords.
- I is an expression that evaluates to an integer.
- CONST1, CONST2 ... are integer literals.
- DoA, DoB, ... DoX are each a line of code or a block of code.
  - If any of them is a block of code, the block needs not to be enclosed by curly brackets { }.
- Arbitrarily many case's are allowed.



- The program compares the value of `I` against the integer value listed in each case in order.
- If the value `I` equals to a listed value, the program exits the **`switch`** statement; otherwise, it checks the next case.
- If no match is found in all cases, the program executes the code listed under **`default`** and then exits the **`switch`** statement.
- When the program exits the **`switch`** statement, it continues with the next line right after the **`switch`** statement.

Note

To implement the desired logic of the switch statement, the `break` statements must be included. For historical reasons, if the `break` statements are not included, the compiler will consider the code syntactically correct. In this case, the code will have a strange behaviour: the code under each case after the true case will be executed.

Highlight

The `switch` statement can only be used to compare integer expression against integer values!

Note

If needed, one can nest if/if-else statement inside a switch statement, or vice versa.

```
#include <stdio.h>
int main()
{
    int x;
    printf("enter 1, 2, or 3\n");
    scanf("%d", &x);
    switch (x)
    {
      case 1:
        printf("You selected option 1\n");
        break;
      case 2:
        printf("You selected option 2\n");
        break;
      case 3:
        printf("You selected option 3\n");
        break;
      default:
        printf("Invalid selection!\n");
    }
    return 0;
}
```

Write a "City Hall" program

Write a program to simulate the following service options and instructions in "City Hall"

Service Option	Instruction
Renew License	Go to Counter 5
Renew Sticker	Go to Counter 8
Pay for Parking Ticket	Go to Counter 2

Use symbolic constants!

Coding Demonstration