CS 31 Discussion

ABDULLAH-AL-ZUBAER IMRAN

WEEK 4: STRINGS AND FUNCTIONS

Discussion Objectives

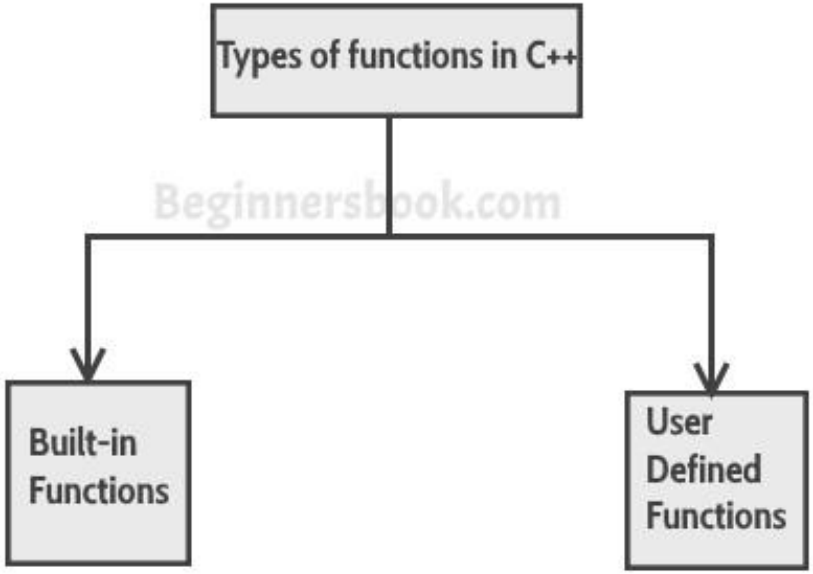
Review and practice things covered during lectures

- Functions
- Coding examples
- Project 3

Programming Challenge

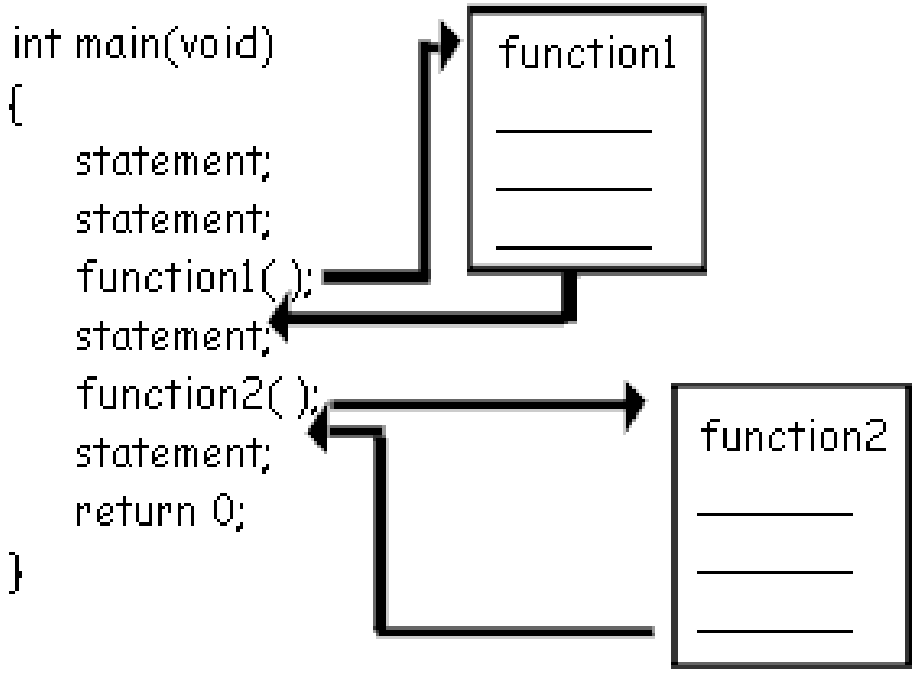
Time for you to ask questions!

Functions



Predefined and precompiled

Declare
Define
Call



Functions (Cont'd)

```
#include <iostream>

// function declaration
return_type func_name (param_1_type param_1_name,
param_2_type param_2_name, ...);

int main() {
    // function call
    return_type var_x = func_name (arg_1, arg_2, ...);
}

/*
 * Note: The top level comment above a function
 * goes here using the multi-line comment, and usually
 * should describe the function's input and output.
 */
return_type func_name (param_1_type param_1_name,
param_2_type param_2_name, ...) {
    // func_name do stuff
}
```

Functions (Cont'd)

```
// function prototype for foo
int foo(int x);

int main() {
    cout << foo(2) << endl;
    cout << foo(0) << endl;
}

// function implementation for foo
int foo(int x) {
    x *= 2;
    if (x < 100)
        return foo(x);
    return x;
}
```

Functions (Cont'd)

//What would be the purpose of this function?

```
void greet(int nTimes, string msg)
        {
                for (int k = 1; k <= nTimes; k++)
                        cout << msg << endl;
        }
```

```
#include <iostream>
using namespace std;

void foo(int x);
int x = 6;

int main() {
    foo(x);
    int x = 5;
    foo(x);

    if (x > 5) {
        int x = 4;
        foo(x);
    } else {
        int x = 3;
        foo(x);
    }
}

void foo(int x) {
    cout << "x = " << x << endl;
}
```

Global scope

func main scope

if-block scope

else-block scope

func foo scope

Built-in Functions

sqrt	Square root	double	double	sqrt(4.0)	2.0	cmath
pow	Powers	double	double	pow(2.0,3.0)	8.0	cmath
abs	Absolute value for int	int	int	abs(-7) abs(7)	7 7	cstdlib
labs	Absolute value for long	long	long	labs(-70000) labs(70000)	70000 70000	cstdlib
fabs	Absolute value for double	double	double	fabs(-7.5) fabs(7.5)	7.5 7.5	cmath
ceil	Ceiling (round up)	double	double	ceil(3.2) ceil(3.9)	4.0 4.0	cmath
floor	Floor (round down)	double	double	floor(3.2) floor(3.9)	3.0 3.0	cmath
exit	End program	int	void	exit(1);	None	cstdlib
rand	Random number	None	int	rand( )	Varies	cstdlib
srand	Set seed for rand	unsigned int	void	srand(42);	None	cstdlib

Functions FAQ

Where do we define functions?

There are two conventional ways, which are equivalent. The requirement is that the function must be defined before it can be used, just like variables.

So you either **completely define it before the function** is used, or add the **prototype** and define it later in the program.

The prototype is a way of telling your compiler that there is such a function, but that we will define it later. Remember to add a semicolon after the prototype, but not after the function header.

Functions FAQ

I defined the function, why doesn't it run?

Defining a function does not imply using it.

You must explicitly call (or invoke) the function somewhere to see it running.

When you call it, it will be run as you defined it.

Where you call it and how you call it depend on you.

Functions FAQ

Why does the return value not show up on the screen?

Because you did not display it, and it's not meant to be displayed.

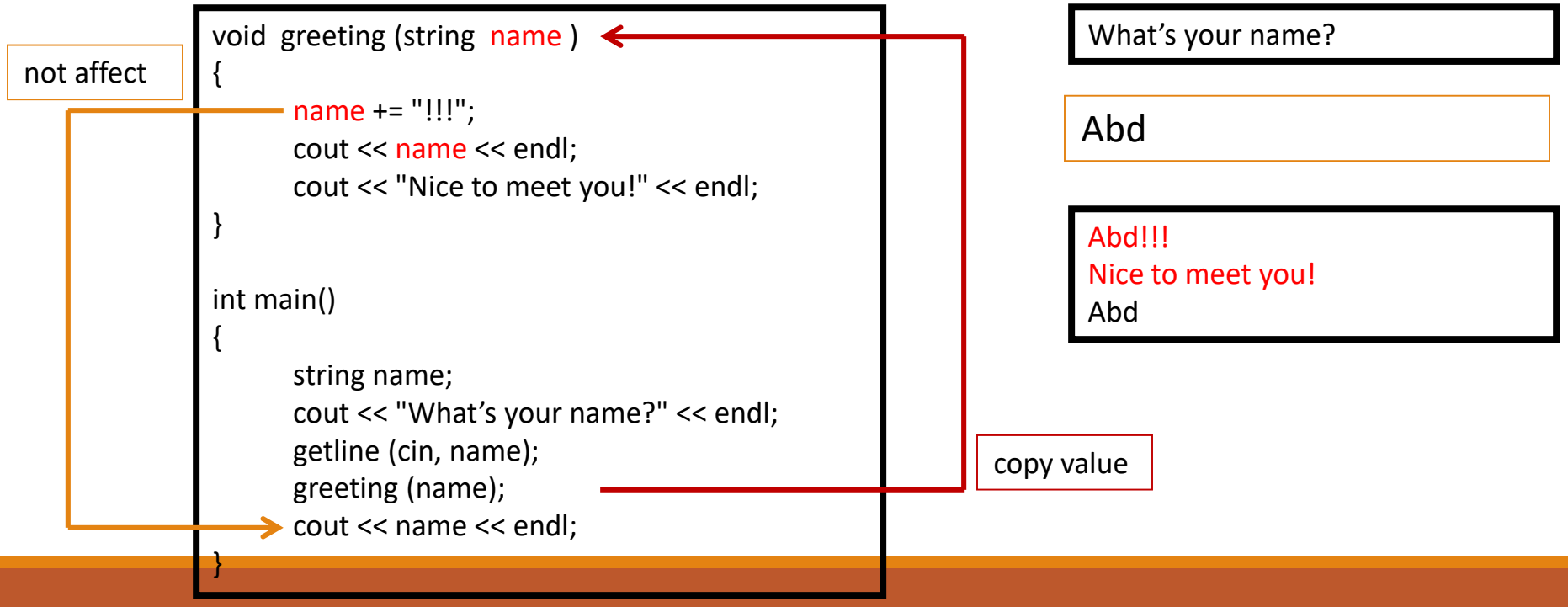
There are people who confusing "returning" with "outputting," which is different.

When you return a value from a function, you return it to whoever called the function.

Passing arguments by value

passing by values into functions

- Doesn't not allow you to access/modify variables outside
- values of arguments exist only in functions



Passing Arguments by Reference

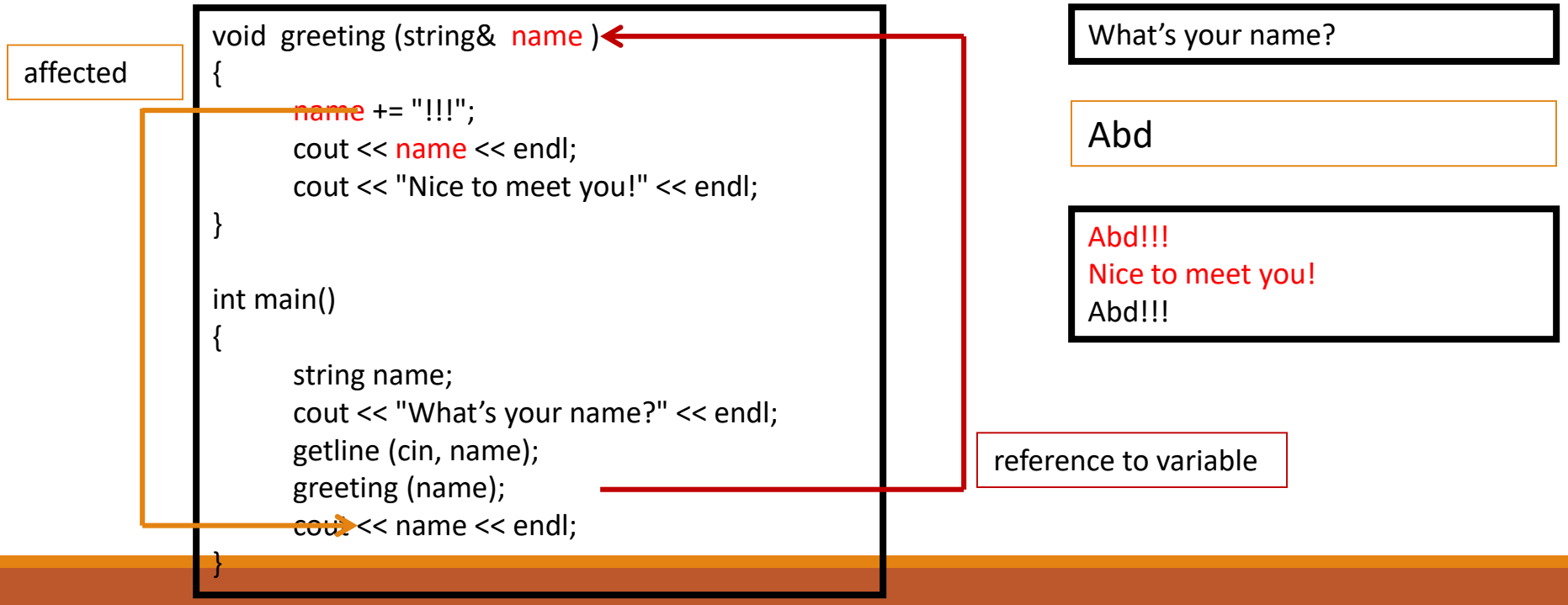
```
#include <iostream>
using namespace std;

void duplicate (int& a, int& b, int& c)
 {
        a*=2;
        b*=2;
        c*=2;
        }

int main ()
 {
        int x=1, y=3, z=7;
        duplicate (x, y, z);

        cout << "x=" << x << ", y=" << y << ", z=" << z;

        return 0;
        }
```

x=2, y=6, z=14

Passing Arguments by Reference

passing in reference to a variable into functions

- allow you to access/modify variables outside



Convert letter to integer

Define a function

```
int convert(char number)
{
    switch (number)
    {
        case '0':
            return 0;
        case '1':
            return 1;
        case '2':
            return 2;
        case '3':
            return 3;
        case '4':
            return 4;
        case '5':
            return 5;
        case '6':
            return 6;
        case '7':
            return 7;
        case '8':
            return 8;
        case '9':
            return 9;
    }
}
```

Convert 1 or 2-digit string to integer

```
int main()
{string str = "12";
int num;

for (size_t i=0; i<str.length(); i++)
{
        if (isdigit(str.at(i)))
        {
                        if (isdigit(str.at(i+1)))
                {
                        num = 10*convert(str.at(i)) + convert(str.at(i+1));
                        i+=1;
                }
                else
                {
                        num = convert(str.at(i));
                }
        }
}
cout << num << endl;
}
```

Incremental Development

- While solving a problem, start thinking from a smallest portion of the problem and try to solve that.
- Incrementally develop the solution for the main problem

Algorithm of incremental development:

1. Start from a simple solution and build on top of that solution.
2. Add a little more
3. Test to make sure the updated version works fine
4. Repeat 2 and 3 until the complete solution covering all possible test cases is implemented

String processing

`#include <string>`

Operation	What it does	Example
`string s = "hello";` `string s = "!!!";`	Declare strings s and s2	
`s.length()` or `s.size()`	Return the length of s	`cout << s.size(); // prints 5`

`s[i]` or `s.at[i]`	Return i-th character. (`i` should be integer between 0 and size-1 (inclusive))	`cout << s[1]; // prints 'e'`
		`cout << s.at(0); // prints 'h'`

`s + s2`	Concatenate two strings	`cout << s + s2; // prints "hello!!!"`

`#include <cctype>`

Operation	What it does
`char c;`	Declare a character c
`isspace(c)`	True if c is a whitespace character
`isalpha(c)`	True if c is a letter
`isdigit(c)`	True if c is a digit
`islower(c)`	True is c is a lowercase letter
`isupper(c)`	True if c is a uppercase letter

String processing

In order to process characters in a string,

E.g., `string str = "123AFb32#@sd";`

```
^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^ ^
```

```
for (int i = 0; i < str.size(); i++) {
  char ch = str[i]; // do something to ch
}
```

for loop

```
int i = 0;
while(i < str.size()){
  char ch = str[i]; // do something to ch
  i++;
}
```

while loop

String processing

Question: count the number of digits and letters in the string `str`.

str	#digit	#letter
“ABC12@cd”	2	5
“sd#12#12”	4	2

Operation	What it does
`char c;`	Declare a character c
`isspace(c)`	True if c is a whitespace character
`isalpha(c)`	True if c is a letter
`isdigit(c)`	True if c is a digit
`islower(c)`	True is c is a lowercase letter
`isupper(c)`	True if c is a uppercase letter

String processing

Question: given a string, filter out all non-letter characters, and print out the new string which is concatenated by all the letters left.

E.g., `string str = “123AFb32#@sd”;` “Afbsd”

```
string concatLetter(string str);
```

```
#include <string>
```

Operation	What it does	Example
`s + s2`	Concatenate two strings	`cout << s + s2; // prints “hello!!!”`

```
#include <cctype>
```

Operation	What it does
`isalpha(c)`	True if c is a letter
`isdigit(c)`	True if c is a digit

String processing

Question: You are writing a program to filter out the illegal date records in the database, and return the number of legal records in December.

The legal date string:

- `year`(4 digits) `month`(3 letters, all UPPERCASE) `day` (1/2 digits).

The `month` is guaranteed to be all uppercase letters.

Only care about number of characters!

```
int filterCount(string str);
```

str	Y/N
1993DEC3	Y
2004DEC52	Y
12MAR3	N
2012AU15	N
2016OCT2	N

2

Characters and Integers

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

ASCII code

char	int
'0'	48
'1'	49
'2'	50
'3'	51
'4'	52
'5'	53
'6'	54
'7'	55
'8'	56
'9'	57

'0' is not mapped to 0!
However, the integer code for chars '0' through '9' are contiguous.

Characters and Integers

char	int
‘0’	48
‘1’	49
‘2’	50
‘3’	51
‘4’	52
‘5’	53
‘6’	54
‘7’	55
‘8’	56
‘9’	57

```
char ch = '0';
ch++; // ch is '1'
int a = ch - '0'; // a is 1
ch += 7; // ch is '8'
a = ch - '0'; // a is 8
```

Characters and Integers

Question: Given a string `str` which contains several digits (0-9), how do you derive the integer value that it represents?

```
int cast(string str)
```

str	return value
“123”	123
“45”	45

Hint:

- ‘4’ -> int a = ‘4’ – ‘0’; // a is 4
- ‘5’ -> int a = ‘5’ – ‘0’; // a is 5
- 45 = 4*10 + 5

Project3: Three Functions

bool isCourseWellFormed(string course)→returns true if its parameter is a syntactically valid course, and false otherwise. Syntactically valid courses: N2eE01n0e2e1 and W42. Syntactically invalid courses: 3sn, e1x, N144, and w2+n3.

---

int driveSegment(int r, int c, char dir, int maxSteps)→determines the number of steps a car starting at position (r,c) could travel in the direction indicated by dir. In the normal case, when this function is called, (r,c) is an empty grid position, dir is one of the letters N, E, S, or W, in either upper or lower case, and maxSteps is the proposed number of steps to travel in the indicated direction.

int driveCourse(int sr, int sc, int er, int ec, string course, int& nsteps)→determines the number of steps a car starting at position (sr,sc) travels when following the indicated course, which should lead to the end position (er,ec). In the normal case, (sr,sc) and (er,ec) are empty grid positions and course is a syntactically valid drivable course. In this case, the function sets nsteps to the number of steps a car starting at (sr,sc) travels when driving the complete course, **and returns 0** if the car ends up at (er,ec), or **1** otherwise. If (sr,sc) or (er,ec) are not valid empty grid positions or if course is not syntactically valid, the function **returns 2** and leaves nsteps unchanged. If (sr,sc) and (er,ec) are empty grid positions and course is syntactically valid, but the car could not drive the complete course without moving to a cell containing a wall or running off the edge of the grid, then the function **returns 3** and sets nsteps to the maximum number of steps that the car can travel along the course (which might be 0). You must not assume that nsteps has any particular value at the time this function is entered.

Project3: What you will turn in

1. A text file named **maze.cpp** that contains the source code for your C++ program
2. A file named **report.docx** or **report.doc** (in Microsoft Word format) or **report.txt**
   a) A brief description of notable obstacles you overcame.
   b) A description of the design of your program (pseudocode)
   c) A list of the test data that could be used to thoroughly test your program, along with the **reason** for and the expected result of each test.

Time due: 9:00 PM Monday, May 6

Pseudocode

Pseudocode is usually a more effective means of communicating an algorithm than a narrative paragraph. It should not be merely a statement-by-statement rephrasing of the code.

Examples of PSEUDOCODE

```
1.  if (score is greater than or equal to 90)
        grade is A
2.  if (hours worked are less than or equal to 40)
        wages = rate * hours
    Otherwise
        wages = (rate * 40) + 1.5 * (rate * (hours -40))
3.  if (temperature is greater than 20 degrees and it is not raining)
        go to play golf!
```

Equivalent C++ code:

```
1.  if (score >= 90)
        cout << "Grade: " << 'A' << endl;
2. if (Hours <= 40)
        wages = rate * Hours;
   else if (Hours > 40)
        wages = (rate*40) + 1.5 * (rate * (Hours -40));
3. if ((temp > 20) && (!(raining))
        cout << "go out to play golf! ";
```

Thanks!

Questions?

Some of the materials presented have been taken from other TA discussions