String Functions (page 1/7)

Note: This 7-page table will be provided on Test #3

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; int j; S0 = "0123456789012345678 9"; S1 = "Intro to Programming"; S2 = "Program"; S3 = "o"; S4 = "aeiou";

find	S1.find(S2)
	S1.find(S2,P)
		Returns the first occurrence of S2 within S1 beginning at position P (at P=0 if omitted). Returns NPOS (-1) if not found.	j = S1.find(S2); // j = 9
			j = S1.find(S3,6); // j = 7
			j = S1.find("p"); // j = -1

rfind	S1.rfind(S2)
	S1.rfind(S2,P)
		Returns the last occurrence of S2 within S1 beginning at position P (at end of S1 if omitted). Returns NPOS (-1) if not found.	j = S1.rfind(S2); // j = 9
			j = S1.rfind(S3,6); // j = 4

find_first_of	S1.find_first_of(S2)
	S1.find_first_of(S2,P)
		Returns the first occurrence of any character in S2 within S1 beginning at position N (at P=0 if omitted). Returns NPOS (-1) if not found.	// find first vowel in S1 (lower case)
			j = S1.find_first_of (S4); // j = 4
			j = S1.find_first_of (S4,9); // j = 11
			j = S1.find_first_of (S4,18); // j = -1

String Functions (page 2/7)

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; int j; S0 = "0123456789012345678 9"; S1 = "Intro to Programming"; S2 = "Program"; S3 = "o"; S4 = "aeiou";

find_first_not_of	S1.find_first_not_of(S2)
	S1.find_first_not_of(S2,P)
		Returns the first occurrence of any character in S2 that is not in S1 beginning at position P (at P=0 if omitted). Returns NPOS (-1) if not found.	// find first character in S1 not a vowel
			j = S1.find_first_not_of (S4); // j = 0
			j = S1.find_first_not_of (S4,7); // j = 8
			j = S1.find_first_not_of (S4,18);// j=18

find_last_of	S1.find_last_of(S2)
	S1.find_last_of(S2,P)
		Returns the last occurrence of any character in S2 that is in S1 beginning at position P (at end of S1 if omitted). Returns NPOS (-1) if not found.	// find last vowel in S1
			j = S1.find_last_of (S4); // j = 17
			j = S1.find_last_of (S4,9); // j = 7
			j = S1.find_last_of (S4,3); // j = -1

String Functions (page 3/7)

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; int j; S0 = "0123456789012345678 9"; S1 = "Intro to Programming"; S2 = "Program"; S3 = "o"; S4 = "aeiou";

find_last_not_of	S1.find_last_not_of(S2)
	S1.find_last_not_of(S2,P)
		Returns the last occurrence of any character in S2 that is not in S1 beginning at position P (at end of S1 if omitted). Returns NPOS (-1) if not found.	// find last character in S1 not a vowel
			j = S1.find_last_not_of (S4); // j = 19
			j = S1.find_last_not_of (S4,11); // j=10
			j = S1.find_last_not_of (S4,0); // j= 0

substr	S5 = S1.substr(P)
	S5 = S1.substr(P,N)
		Returns a string N characters in length, beginning at position P. If N is omitted, returns a string from P to the end of string S1.	S5 = S1.substr(9); //S5="Programming"
			S5 = S1.substr(9,4); // S5 = "Prog"

String Functions (page 4/7)

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; S0 = "0123456789012345678 9"; S1 = "Computer"; S2 = "I Like Programming";

append	S1.append(S2)
	
	S1.append(S2,P,N)
		Appends a string of N characters beginning at position P in S2 onto the end of S1. If P and N are omitted, S2 is appended to the end of S1.	S3=S1;
			S3.append(S3); // S3 = "ComputerComputer“
			S3=S1;
			S3.append(S2,6,8); // S3 = "Computer Program"

assign	S1.assign(S2,P,N)	Assigns (replaces) the entire string S1 with N characters of S2 beginning at position P.	S3.assign(S2,7,7); // S3 = "Program"

erase	S1.erase(P,N)	Removes N characters of S1 beginning at position P.	S3=S1;
			S3.erase(7,1); // S3 = "Compute"
			S3=S2;
			S3.erase(0,7); // S3= "Programming"

insert	S1.insert(P,S2)	Inserts S2 into S1 beginning at position P.	S4=S2; S4.insert(7,S1); // S4 = "I like ComputerProgramming"

String Functions (page 5/7)

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; S0 = "0123456789012345678 9"; S1 = "Computer"; S2 = "I Like Programming";
push_back	S1.push_back(C1)	Appends a single character C1 to S1.	char C4='s'; S3=S1; S3.push_back(C4); // S3= "Computers"
replace	S1.replace(P,N,S2)	Replaces N characters in S1 beginning in position P with S2.	S4="C++"; S3=S2; S3.replace(7,11,S4); // S3= "I Like C++"

resize	S1.resize(N)
	S1.resize(N,C)
		Changes size of S1 to have N characters. If made longer, filled to end with character C. If made shorter, truncated.	S3=S1;
			S3.resize(4); // S3= "Comp"
			S3=S1;
			S3.resize(13, '*'); // S3= "Computer****"

swap	S1.swap(S2)	Swaps the contents of S1 and S2.	S3="First";
			S4="Second";
			S3.swap(S4); // S3= "Second"
			// S4= "First"

String Functions (page 6/7)

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; S0 = "0123456789012345678 9"; S1 = "Computer"; S2 = "I Like Programming";

compare	S1.compare(P)
	
	S1.compare(P,N,S2)
		Compares N characters of S1 beginning at position P with S2. Return value is:
		0 if S1 = S2
		>0 if S1>S2
		<0 if S1<S2
			S3="camp";
			S4="comp";
			S5="computer";
			S3.compare(S4); // returns -1 since S3<S4
			S4.compare(S3); // returns 1 since S3>S4
			S4.compare(S4); // returns 0 since S3==S4
			if(!(S5.compare(0,4,S4))) cout << "First 4 same";
			// Output: "First 4 same"

empty	S1.empty()	Returns 1 if S1 is empty, otherwise 0.	S1.empty(); // 0 (False empty)
			S3 = "";
			if (S3.empty()) cout << "S3 empty"; // S3 empty

length	S1.length()	Returns number of characters in S1.	S2.length(); // 18
size	S1.size()	Same as length().	S2.size(); // 18
at	S1.at(P)	Used to access the character at position P in S1.	S2.at(4); // 'k'
[ ]	S1[P]	Same as S1.at(P)	S2[4]; // 'k'

String Functions (page 7/7)

Function	Form(s)	Description – based on the Form(s) shown	Examples. In each case use: string S0,S1,S2,S3,S4,S5; S0 = "01234567890123456789"; S1 = "Computer"; S2 = "I Like Programming";

capacity	S1.capacity()	Returns capacity of S1 (same as size unless increased using reserve( )?)	S2.capacity(); // 18
			S2.reserve(50);
			S2.capacity(); // 50
			S2.size(); // 18

c_str	S1.c_str()	Converts string S1 to a character array.	// use c_str( ) with variable file names since
			// fstream wants character arrays
			string FileName = "MyFile.txt";
			ifstream(FileName); // may yield compiler error
			ifstream(FileName.c_str()); // fixes the error

tolower	tolower(S1[P])	Converts character at position P from upper case to lower case (ignored if not an upper case letter)	S2[6] = tolower(S2[6]); // blank ignored
			S2[7] = tolower(S2[7]); // convert P to p
			S2[8] = tolower(S2[8]); // r ignored
			// now S2 = "I Like programming";

Note that the following functions were introduced earlier in the course for working with characters. They also work with strings (see example above using **tolower( )**).

tolower, toupper

islower, isupper, isdigit, isspace, isblank, isalpha (alphabetic), **isalnum** (alphanumeric)

ASCII Code

TABLE 3.5 — Standard C++ characters and their ASCII codes/values (in decimal)

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