Swinburne University Of Technology

Faculty of Science, Engineering and Technology

LABORATORY COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Lab number and title: 2, Basic I/O
Lecturer: Dr. Markus Lumpe

A journey of a thousand miles begins with a single step.

Lao Tsu

The ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
00	00	NUL	32	20	SP	64	40	@	96	60	`
01	01	SOH	33	21	!	65	41	A	97	61	a
02	02	STX	34	22	"	66	42	B	98	62	b
03	03	ETX	35	23	#	67	43	C	99	63	c
04	04	EOT	36	24	$	68	44	D	100	64	d
05	05	ENQ	37	25	%	69	45	E	101	65	e
06	06	ACK	38	26	&	70	46	F	102	66	f
07	07	BEL	39	27	'	71	47	G	103	67	g
08	08	BS	40	28	(	72	48	H	104	68	h
09	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Basic I/O in C++

The goal of this laboratory session is to develop a small Win32 console application that counts the occurrences of each printable non-whitespace character in a given input text stream. The application consists of two parts: a class `CharacterCounter` and a `main` function that drives the counting process.

The class `CharacterCounter` is specified as follows:

```
#pragma once

#include <iostream>

class CharacterCounter
{
private:
  int fTotalNumberOfCharacters;
  int fCharacterCounts[256];                    // We count all 256 byte values

public:
  CharacterCounter();

  void count( unsigned char aCharacter );

  friend std::ostream& operator<<( std::ostream& aOStream,
                                   const CharacterCounter& aCharacterCounter );
};
```

The class `CharacterCounter` records the total number of counted characters and the frequencies of those characters. The class has one constructor to properly initialize the data members, a `count` method that takes a character (an `unsigned char` value), and declares the stream output `operator<<` for `CharacterCounter` as a friend of class `CharCounter`.

The implementations of the constructor and the `count` method are straightforward. The constructor initializes all data members with 0, whereas `count` has to increment the corresponding data members.

The **`operator<<`** is a bit more complex. First, the **`operator<<`** should only print those characters that actually occur in the input text stream (i.e., you need to filter the 0s). Second, you need to use a simple trick to print an integer value as a character value. You can achieve this using the cast operator `(type)value`. For example, if you program defines an integer variable `lIntValue`, then `(char)lIntValue` yields a character value, possibly shorten the value to range between 0 and 255.

The `main` function declares an object of type `CharacterCounter`, an `unsigned char` variable, and performs the counting process:

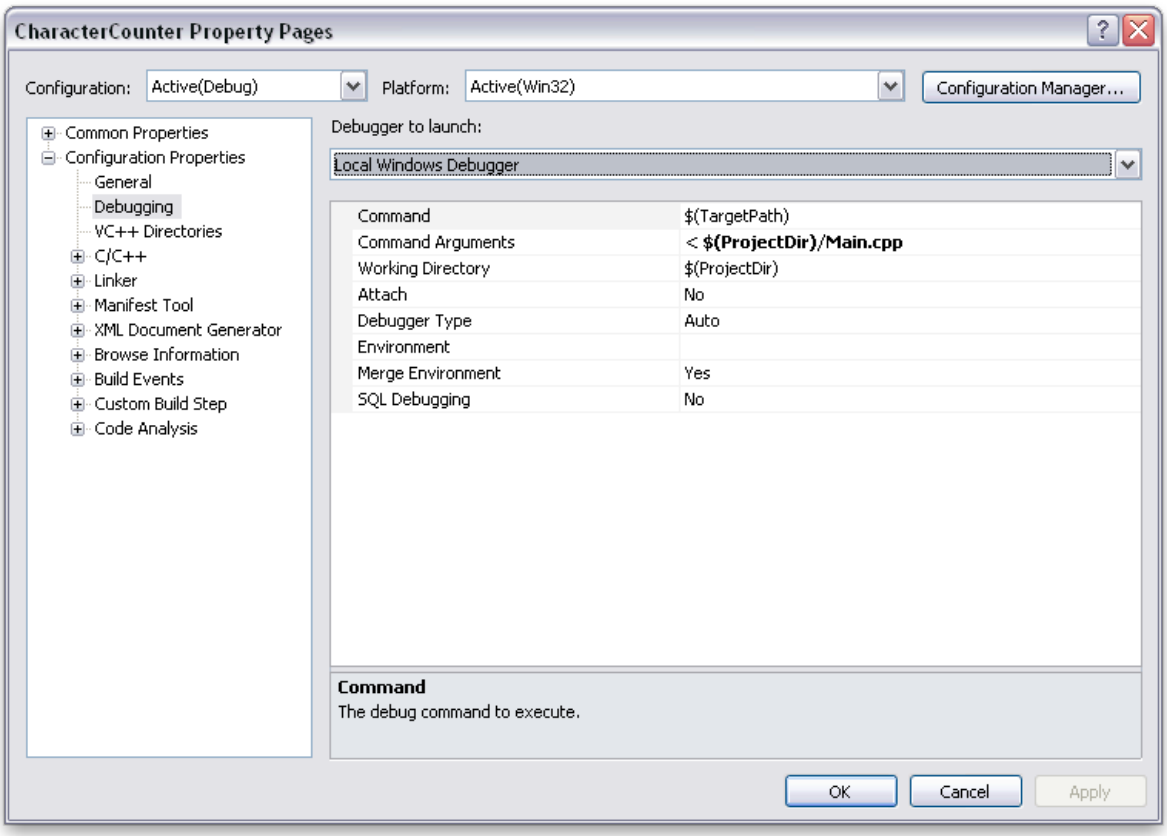
```
CharacterCounter lCounter;

unsigned char lChar;

while ( cin >> lChar )
{
    lCounter.count( lChar );
}

cout << lCounter;
```

You can use the implementation file (i.e., .cpp file) of your project as input. If the name of that file is `Main.cpp`, then you can use the following setting of the Command Arguments:



Running the program (Start Without Debugging) produces an output similar to the following sample:

This exercise requires approximately 70 lines of low-density C++ code.