Misr International University (MIU)

Faculty of Computer Science

Computer Science Program

CSC 105: Introduction to Programming & Problem Solving

Dr. Ayman Ezzat and Dr. Ashraf AbdelRaouf

Lecture 3

The character, string data Types
Files

char Data Type

- The smallest integral data type
- Used for single characters: letters, digits, and special symbols
- Each character is enclosed in single quotes
  - `'A', 'a', '0', '*', '+', '$', '&'`
- A blank space is a character
  - Written `' '`, with a space left between the single quotes

char Data Type (cont'd.)

- Different character data sets exist
- ASCII: American Standard Code for Information Interchange
  - Each of 128 values in ASCII code set represents a different character
  - Characters have a predefined ordering based on the ASCII numeric value

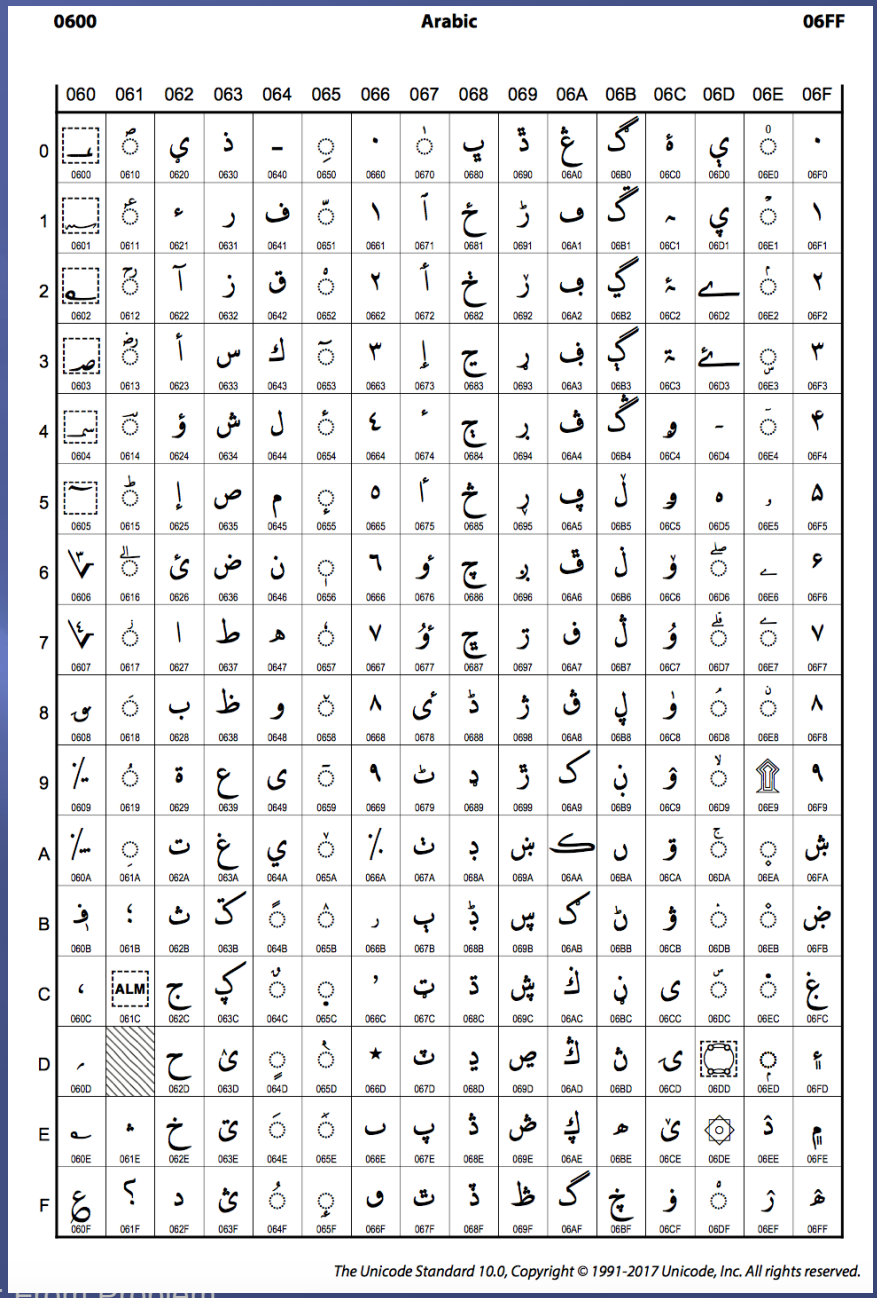
ASCII Character set

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	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	Neg. 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	End 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	Substitution	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	□

ASCII Character set

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Unicode Arabic Character set



Comparing Character Type

Table 4-2 Evaluating Expressions Using Relational Operators and the ASCII Collating Sequence

Expression	Value of Expression	Explanation
`' ' < 'a'`	`true`	The ASCII value of `' '` is 32, and the ASCII value of `'a'` is 97. Because `32 < 97` is `true`, it follows that `' ' < 'a'` is `true`.
`'R' > 'T'`	`false`	The ASCII value of `'R'` is 82, and the ASCII value of `'T'` is 84. Because `82 > 84` is `false`, it follows that `'R' > 'T'` is `false`.
`'+' < '*'`	`false`	The ASCII value of `'+'` is 43, and the ASCII value of `'*'` is 42. Because `43 < 42` is `false`, it follows that `'+' < '*'` is `false`.
`'6' <= '>'`	`true`	The ASCII value of `'6'` is 54, and the ASCII value of `'>'` is 62. Because `54 <= 62` is `true`, it follows that `'6' <= '>'` is `true`.

The string Type

- To use the data type string, the program must include the header file <string>
- The statement:

```
string name = "William Jacob";
```

  declares name to be a string variable and also initializes name to "William Jacob"
- The first character, 'W', in name is in position 0, the second character, 'i', is in position 1, and so on

The string Type (continued)

- The variable name is capable of storing any size string
- Binary operator + (to allow the string concatenation operation), and the array subscript operator [], have been defined for the data type string
- For example, If str1 = "Sunny", the statement stores the string "Sunny Day" into str2:

```
str2 =  str1 + " Day";
```

string Comparison Example

- Suppose we have the following declarations:
  - string str1 = "Hello";
  - string str2 = "Hi";
  - string str3 = "Air";
  - string str4 = "Bill";

Table 4-3 Evaluating Logical Expressions with String Variables

Expression	Value
`str1 < str2`	`true`
`str1 > "Hen"`	`false`
`str3 < "An"`	`true`
`str1 == "hello "`	`false`
`str3 <= str4`	`true`

length Function

- Length returns the number of characters currently in the string
- The syntax to call the length function is:

  strVar.length()

  where strVar is variable of the type string
- length has no arguments
- length returns an unsigned integer
- The value returned can be stored in an integer variable

find Function

- find searches a string for the first occurrence of a particular substring
- Returns an unsigned integer value of type string::size_type giving the result of the search
- The syntax to call the function find is:

  strVar.find(strExp)

  where strVar is a string variable and strExp is a string expression evaluating to a string
- The string expression, strExp, can also be a character

find Function (continued)

- If successful, find returns the position in strVar where the match begins
- For the search to be successful the match must be exact
- If unsuccessful, find returns the special value string::npos (“not a position within the string”)
- To find next, strVar.find(strExp, Pos), where the Pos is the position to start searching with.

Append Function

- Append returns the string after concatenating with other string
- The syntax to call the length function is:

  strVar.append(strVar1)

  where strVar, strVar1 are variables of type string
- Append can also be used to concatenate sub-string of other string in a certain position
- The syntax to call the length function is:

  strVar.append(strVar1, 5, 7)

  where Append strVAR1 from the fifth character and append 7 characters

substr Function

- substr returns a particular substring of a string
- The syntax to call the function substr is:

  strVar.substr(expr1,expr2)

  The expression expr1 specifies a position within the string (starting position of the substring)

  The expression expr2 specifies the length of the substring to be returned

Replace Function

- Replaces part of the string with another string
- string& replace (size_t *pos*, size_t *len*, const string& str);
  - St1.replace(5, 9, St2); put 9 characters from St2 on the 5th position from St1.

compare Function

- comapre returns an int represent the comparison between two strings
- The syntax to call the function substr is:

```
int comp=strVar.compare(strVar1)
```

Where comp is zero if the two strings are equal

\+ if the strVar is bigger than strVar1, else is negative

The value of comp depends on the distance between the two strings

swap Function

- swap interchanges the contents of two string variables
- The syntax to use the function swap is

```
strVar1.swap(strVar2);
```

  where strVar1 and strVar2 are string variables
- Suppose you have the following statements:

```
string str1 = "Warm";
string str2 = "Cold";
```

- After str1.swap(str2); executes, the value of str1 is "Cold" and the value of str2 is "Warm"

I/O and the `string` Type

- An input stream variable (`cin`) and extraction operator >> can read a string into a variable of the data type string
- Extraction operator
  - Skips any leading whitespace characters and reading stops at a whitespace character
  - Should not be used to read strings with blanks
- The function `getline`
  - Reads until end of the current line
  - Should be used to read strings with blanks

File Input/Output

- File: area in secondary storage to hold info
- File I/O
  1. Include `fstream` header
  2. Declare file stream variables
  3. Associate the file stream variables with the input/output sources
  4. Use the file stream variables with `>>`, `<<`, or other input/output functions
  5. Close the files File Input/Output

File Input/Output

- The type is a bitmask type that describes an object that can store the opening mode for several iostreams objects. The distinct flag values (elements) are:

- **app**, to seek to the end of a stream before each insertion.
- **ate**, to seek to the end of a stream when its controlling object is first created.
- **binary**, to read a file as a binary stream, rather than as a text stream.
- **in**, to permit extraction from a stream.
- **out**, to permit insertion to a stream.
- **trunc**, to delete contents of an existing file when its controlling object is created.

EOF-Controlled while Loops

- Use an EOF (End Of File)-controlled while loop
- The logical value returned by cin can determine if the program has ended input
- The syntax is:

```
cin >> variable;
while (cin)
{
        .
        cin >> variable;
        .
}
```

The eof Function

- The function eof can determine the end of file status
- Like other I/O functions (get, ignore, peek), eof is a member of data type istream
- The syntax for the function eof is:

  istreamVar.eof()

  where istreamVar is an input stream variable, such as cin

File Input stream

- File: open, seekg, eof, close,

```cpp
// basic_istream_seekg.cpp
  // compile with: /EHsc
  #include <iostream>
  #include <fstream>

  int main ( )
  {
     using namespace std;
     ifstream file;
     char c, c1;

     file.open( "basic_istream_seekg.txt" );
     file.seekg(2);   // seek to position 2
     file >> c;
     cout << c << endl;
  }
```

```cpp
// basic_ios_eof.cpp
  // compile with: /EHsc
  #include <iostream>
  #include <fstream>

  using namespace std;

  int main( int argc, char* argv[] )
  {
     fstream   fs;
     int n = 1;
     fs.open( "basic_ios_eof.txt" );   // an empty file
     cout << boolalpha
     cout << fs.eof() << endl;
     fs >> n;   // Read the char in the file
     cout << fs.eof() << endl;
  }
```

Example

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

int main() {
    ifstream inFile;
    ofstream outFile;
    double exam1, exam2, exam3, average;

    inFile.open("e:\\grades.dat");
    outFile.open("e:\\average.dat");
    inFile >> exam1 >> exam2 >> exam3;
    average = (exam1 + exam2 + exam3) / 3.0;
    outFile << "Average = " << average << endl;
    inFile.close();
    outFile.close();
    return 0;
}
```

Specifying Input/Output Files at Execution Time

- You can let the user specify the name of the input and/or output file at execution time:

```
ifstream infile;
ofstream outfile;

char fileName[51];     //assume that the file name is at most
                       //50 characters long
```

```
cout << "Enter the input file name: ";
cin >> fileName;

infile.open(fileName);    //open the input file
.
.
.

cout << "Enter the output file name: ";
cin >> fileName;

outfile.open(fileName);   //open the output file
```