String and Character Manipulation
C-style Strings (ntcas)

char name[10] = "Clayton";

C++ Standard Library Strings
(a.k.a. Standard String Class)

#include <string>
using std::string;
...
string name = "Clayton";
std::string

#include <string>
using std::string;

int main()
{
    string name = “Clayton”;
    cout << name.length();
std::string input limitations

string name;
cout << "What’s your name? " ;
cin >> name ;
cout << name << endl ;
std::string name;

```cpp
cout << "What's your name? ";
cin >> name;
cout << name << endl;
```

User enters: Pattie Boyd
std::string input limitations

```cpp
string name;
cout << "What's your name? ";
cin >> name;
cout << name << endl;
```

System outputs: Pattie
std::string input limitations

string name;
string hometown;
cout << "What’s your name? ";
cin >> name;
cout << "What is your hometown?";
cin >> hometown;
std::string input limitations

```cpp
string name;
string hometown;

cout << "What's your name? ";
cin >> name;

cout << "What is your hometown?"; cin >> hometown;
```

System outputs: What's your name?
std::string input limitations

string name;
string hometown;
cout << "What's your name? ";
cin >> name;
cout << "What is your hometown?";
cin >> hometown;

User enters: James Marshall
std::string input limitations

string name;
string hometown;
cout << "What’s your name? ";
cin >> name;
cout << "What is your hometown?"; cin >> hometown;

System places “James Marshall” into the input buffer
std::string input limitations

string name;
string hometown;
cout << "What’s your name? ";
cin >> name;
cout << "What is your hometown? ";
cin >> hometown;

System pulls “James” from the buffer and into name
std::string input limitations

```cpp
string name;
string hometown;
cout << "What's your name? ";
cin >> name;
cout << "What is your hometown?";
cin >> hometown;
```

System outputs: What is your hometown?
std::string input limitations

```cpp
string name;
string hometown;
cout << "What’s your name? ";
cin >> name;
cout << "What is your hometown? ";
cin >> hometown;
```

System does not wait for the user input
std::string input limitations

```cpp
string name;
string hometown;
cout << "What’s your name? ";
cin >> name;
cout << "What is your hometown? “;
cin >> hometown;
```

Instead “Marshall” is pulled from the buffer and into hometown
std::string input fix

// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What's your name? ";
getline(cin, name, '\n');
cout << "What is your hometown?";
getline(cin, hometown, '\n');
cout << "Name: " << name;
cout << "Hometown: " << hometown;
```cpp
// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;

cout << "What’s your name? ";
getline(cin, name, ‘\n’);

cout << "What is your hometown?";
getline(cin, hometown, ‘\n’);

cout << "Name: “ " << name;
cout << "Hometown: “ " << hometown;
```
### std::string input fix

```cpp
// getline(input_stream, string_var, delimiter_char);
string name;
string hometown;
cout << "What's your name? ";
getline(cin, name, '"n');
cout << "What is your hometown?";
getline(cin, hometown, '"n');
cout << "Name: " << name;
cout << "Hometown: " << hometown;
```
std::string input fix

// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << "What is your hometown?";
getline(cin, hometown, ‘\n’);
cout << "Name: " << name;
cout << "Hometown: " << hometown;
```cpp
// getline(input_stream, string_var, delimiter_char);
string name;
string hometown;
cout << "What's your name? ";
getline(cin, name, '\n');
cout << "What is your hometown?";
getline(cin, hometown, '\n');
cout << "Name: " << name;
cout << "Hometown: 
```

Delimiter Character is NOT included in input; it’s discarded.
```cpp
std::string input fix

// getline(input_stream, string_var, delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << "What is your hometown?";
getline(cin, hometown, ‘\n’);
cout << "Name: “ << name;
cout << "Hometown: “ << hometown;
```

System outputs: What’s your name?
std::string input fix

// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << "What is your hometown?";
ggetline(cin, hometown, ‘\n’);
cout << "Name: " << name;
cout << "Hometown: " << hometown;

User enters: Janis Joplin
// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << "What is your hometown?";
getline(cin, hometown, ‘\n’);
cout << "Name: " << name;
cout << "Hometown: " << hometown;

System outputs: What is your hometown?
std::string input fix

// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << “What is your hometown?”;
gidine(cin, hometown, ‘\n’);
cout << “Name: “ << name;
cout << “Hometown: “ << hometown;

User enters: Port Arthur
```cpp
std::string input fix

// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << "What is your hometown?";
getline(cin, hometown, ‘\n’);
cout << "Name: " << name;
cout << "Hometown: " << hometown;
```

System outputs: Janis Joplin
// getline(input_stream,string_var,delimiter_char);
string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name, ‘\n’);
cout << "What is your hometown?";
getline(cin, hometown, ‘\n’);
cout << "Name: " << name;
cout << "Hometown: " << hometown;
System continues output: Janis JoplinPort Arthur
std::string alternative fix

string name;
string hometown;
cout << "What’s your name? ";
getline(cin, name);
cout << "What is your hometown?";
getline(cin, hometown);
cout << "Name: " << name;
cout << "Hometown: " << hometown;
std::string more input nuances

string name;
int age;
cout << “What’s your age? ”;
cin >> age;
cout << “What’s your name? ”;
getline(cin, name);
std::string more input nuances

string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
getline(cin, name);

System outputs: What’s your age?
std::string more input nuances

string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
 getline(cin, name);

User enters: 8
std::string more input nuances

string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
getline(cin, name);

System places “8\n” into the input buffer
std::string more input nuances

string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
getline(cin, name);

System pulls “8” from the buffer and into age
std::string more input nuances

```cpp
string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
getline(cin, name);
```

System outputs: What’s your name?
std::string more input nuances

string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
getline(cin, name);

System does not wait for the user input
std::string more input nuances

string name;
int age;
cout << "What’s your age? ";
cin >> age;
cout << "What’s your name? ";
getline(cin, name);

Instead it pulls "\n" from the buffer into name and getline is finished
std::string vs C-string(ntca)

```cpp
string sname;
const int SIZE = 80;
char cname[SIZE];

cout << "What's your given name? ";
getline(cin, sname);
cout << "What's your family name? ";
cin.getline(cname, SIZE-1);  //different syntax!!
```
more C-string getline

const int SIZE = 10;
char name[SIZE];

cout << "What's your name? ";
cin.ignore(500, '\n');
cin.getline(name, SIZE-1);
cout << name << endl;
more C-string getline

```cpp
const int SIZE = 10;
char name[SIZE];

cout << "What’s your name? \n";
cin.ignore(500, '\n');
cin.getline(name, SIZE-1);
cout << name << endl;
```
more C-string getline

const int SIZE = 10;
char name[SIZE];

cout << "What’s your name? ";
cin.ignore(500, ‘\n’);
cin.getline(name, SIZE-1);
cout << name << endl;

System outputs: What’s your name?

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
more C-string getline

```cpp
const int SIZE = 10;
char name[SIZE];

cout << "What's your name? ";
cin.ignore(500, '\n');
cin.getline(name, SIZE-1);
cout << name << endl;
```

System clears any extraneous \n from the input buffer
more C-string getline

const int SIZE = 10;
char name[SIZE];

cout << “What’s your name? ”;
cin.ignore(500, ‘\n’);
cin.getline(name, SIZE-1);
cout << name << endl;

User enters: Chimley

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>h</td>
<td>i</td>
<td>m</td>
<td>l</td>
<td>e</td>
<td>y</td>
<td>\0</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
more C-string getline

```cpp
const int SIZE = 10;
char name[SIZE];

cout << "What's your name? ";
cin.ignore(500, '\n');
cin.getline(name, SIZE-1);
cout << name << endl;
```

System outputs: Chimley

```
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
  C    h    i    m    l    e    y \0   ?  ?
```
Extraction vs. getline

- The extraction operator (cin>>variable) will always skip leading whitespace and leave \n in the buffer.
- The getline functions will read leading whitespace and will not leave the \n in the buffer.

**Rule:** always know what is in your input buffer.
#include<ctype>

- **toupper(char)**
  - returns the uppercase of arg sent `toupper('a');` -> 'A'
- **tolower(char)**
  - similar
- **isupper(char)**
  - returns bool: true if uppercase `isupper('a');` -> false
- **islower(char)**
  - similar
- **isalpha(char)**
  - similar
- **isdigit(char)**
  - similar
- **ispunct(char)**
  - returns bool: true if punctuation `ispunct('!');` -> true
- **isspace(char)**
  - returns bool: true if whitespace - space, newline, tab
C-string Manipulation

```c
int i = 0, count = 0;
char ntca[20] = "Hello! Hi.";

while (ntca[i] != '\0')
{
    if (ispunct(ntca[i]))
        count++;
    i++;
}

cout<<count<<endl;
```
C-string Manipulation

```cpp
int i = 0, count = 0;
char ntca[20] = "Hello! Hi.";

while (ntca[i] != '\0')
{
    if (isspace(ntca[i]))
        count++;

    i++;
}

cout << count << endl;
```
C-string Manipulation

```cpp
int i = 0, count = 0;
char ntca[20] = "Hello! Hi."

while (ntca[i] != '\0')
{
    ntca[i] = toupper(ntca[i]);
    i++;
}

cout << ntca << endl;
```
HELLO! HI.
Write Your Own

```cpp
bool IsDigit (const char input)
```

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Name</th>
<th>Char</th>
<th>Ctrl-char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Null</td>
<td>NUL</td>
<td>CTRL-@</td>
<td>32</td>
<td>20</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>'</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Start of heading</td>
<td>SOH</td>
<td>CTRL-A</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Start of text</td>
<td>STX</td>
<td>CTRL-B</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>End of text</td>
<td>ETX</td>
<td>CTRL-C</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>End of xmit</td>
<td>EOT</td>
<td>CTRL-D</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Enquiry</td>
<td>ENQ</td>
<td>CTRL-E</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Acknowledge</td>
<td>ACK</td>
<td>CTRL-F</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Bell</td>
<td>BEL</td>
<td>CTRL-G</td>
<td>39</td>
<td>27</td>
<td></td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Backspace</td>
<td>BS</td>
<td>CTRL-H</td>
<td>40</td>
<td>28</td>
<td></td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Horizontal tab</td>
<td>HT</td>
<td>CTRL-I</td>
<td>41</td>
<td>29</td>
<td></td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Line feed</td>
<td>LF</td>
<td>CTRL-J</td>
<td>42</td>
<td>2A</td>
<td></td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Vertical tab</td>
<td>VT</td>
<td>CTRL-K</td>
<td>43</td>
<td>2B</td>
<td></td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Form feed</td>
<td>FF</td>
<td>CTRL-L</td>
<td>44</td>
<td>2C</td>
<td></td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Carriage feed</td>
<td>CR</td>
<td>CTRL-M</td>
<td>45</td>
<td>2D</td>
<td></td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Shift out</td>
<td>SO</td>
<td>CTRL-N</td>
<td>46</td>
<td>2E</td>
<td></td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Shift in</td>
<td>SI</td>
<td>CTRL-O</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Data line escape</td>
<td>DLE</td>
<td>CTRL-P</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Device control 1</td>
<td>DC1</td>
<td>CTRL-Q</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Device control 2</td>
<td>DC2</td>
<td>CTRL-R</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Device control 3</td>
<td>DC3</td>
<td>CTRL-S</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Device control 4</td>
<td>DC4</td>
<td>CTRL-T</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>Neg acknowledge</td>
<td>NAK</td>
<td>CTRL-U</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Synchronous idle</td>
<td>SYN</td>
<td>CTRL-V</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>End of xmit block</td>
<td>ETB</td>
<td>CTRL-W</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Cancel</td>
<td>CAN</td>
<td>CTRL-X</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>End of medium</td>
<td>EM</td>
<td>CTRL-Y</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>Substitue</td>
<td>SUB</td>
<td>CTRL-Z</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>Escape</td>
<td>ESC</td>
<td>CTRL-[</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>{</td>
<td>123</td>
<td>7B</td>
<td>{</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>File separator</td>
<td>FS</td>
<td>CTRL\</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>Group separator</td>
<td>GS</td>
<td>CTRL-]</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>}</td>
<td>125</td>
<td>7D</td>
<td>}</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>Record separator</td>
<td>RS</td>
<td>CTRL^</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>Unit separator</td>
<td>US</td>
<td>CTRL-\</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>\</td>
<td>127</td>
<td>7F</td>
<td>DEL</td>
</tr>
</tbody>
</table>
Write Your Own

bool IsDigit (const char input)
{
  bool digit = false;
  if (input >= 48 && input <= 57)
    digit = true;
  return digit;
}
Write Your Own

bool IsDigit (const char input)
{
    return (input>=48 && input<=57);
}
Character Input and Output

- **get** – allows one character to be read from input
  - `cin.get(char_variable);`
- **peek** – reads the next character from the input buffer without extracting it
  - `char_variable = cin.peek();`
- **putback** – places the character back into the input buffer
  - `cin.putback(char_variable);`
- **put** – outputs a single character
  - `cout.put(char_variable)`
  - equivalent to `cout << char_variable`
get

char next;
cout<<"enter your poem: ";
do
{
    cin.get(next);
    cout<<next;
} while (next != '\n');
get vs. getline

char next;
cout<>"enter your poem: ";
do
{
    cin.get(next);
cout<>next;
} while (next != '\n');

char poetry[500];
cout<>"enter your poem: ";
cin.getline(poetry, 499);
cout<>poetry;
Input Options

- getline()
  - reads line-by-line
- cin >>
  - reads word-by-word
- get()
  - reads character-by-character
End of Session