CS 101: Computer Programming and Utilization

Shivaram Kalyanakrishnan
(Abhiram Ranade's slides, borrowed and edited)
Lecture 17

Today's Lecture

- assert()
- Character arrays

assert()

- Usage: assert(e) where expression e is a boolean.
- If e is true, program execution continues.
- If e is false, program terminates!
- Need #include <cassert> to use this command.
- Can disable with #define NDEBUG

Today's Lecture

- assert()
- Character arrays

Character Arrays

`char name[10], address[50];`

- Defines two arrays, of length 10 and 50 respectively.
- Can be used to store character sequences, or “character strings”.
- An array of length n may store strings of smaller length.

Standard protocol inherited from the C language:

- store characters in the string in the array starting from element 0.
- After all the characters are stored, store the character ‘\0’,
- ‘\0’ = Character whose ASCII value is 0. “null” character.
- Key idea: “Everything until ‘\0’ is a part of the string, not what comes later”
- No need to explicitly store the length of the string.

Character arrays with initialization

```
char n1[20]="Ajanta", n2[]="Ellora";
```

n1 will be created with length 20, and initialized as follows.

- 'A', 'j', 'a', 'n', 't', 'a' will get stored in `n1[0]` through `n1[5]`.
- Finally '\0' will get stored in `n1[6]`.
- The elements `n1[7]` through `n1[19]` will not be initialized.

n2 will be created of length 1 + the length of the string "Ellora".

- It will also be initialized to the string "Ellora" followed by '\0'.

Syntax is only for initialization, not assignment.

Reading into char arrays 1

Example char array: `char buffer[80];`

- First way to read into it:

`cin >> buffer;`

- Reads one word (white space terminated) , stores it into `buffer`, terminated by '\0'.

Suppose user types:

`C++ is nice` (newline terminated)

- Only `C++` would go into `buffer`.

Note: `cin >> buffer` does not mention length of `buffer`, only its starting address.

- If user types more than 80 characters, there will be "index out of range".
  - So allocate a large enough array.
- Note: `cin >> buffer;` works only for char arrays.

Reading into char arrays 2

Example char array: `char buffer[80];`

- Safe way to read into it:

```
cin.getline(buffer,80);
```

- Reads a line (terminated by newline) or at most 79 characters.
- Stores them in buffer, terminated by '\0'. Safe.
- Line read may contain spaces.

Suppose user types

`C++ is nice` (newline terminated)

- `C++ is nice` would go into buffer followed by '\0'.

Printing char arrays

Example char array: `char buffer[80];`

```
... Code to assign value to buffer ...
```

- How to print:

```
cout << buffer;
```

- Print `buffer` content till '\0'.
- **`buffer` assumed to contain a '\0'.**
- Length of array is not important.

Note: For other types of arrays above would print address.

Character array processing

- Usually, the array length is ignored, instead we “process all elements till we find ‘\0’ ”
- The array must contain a ‘\0’.
- Very common idiom for character array processing.

Examples

- Reversing a string
- Checking if string contains letter
- Lexicographic ordering of two strings
- Concatenating two strings