CAAM 420 Daily Note

Hojin Jeon

Nov 11, 2013

1 Passing parameters in given format

```
hojin@hojin-VirtualBox:~$ ./a.out 10 5 240
```

Most of the projects given so far used parameters passed by values as above. Despite being simple and easy, such programming method does not give much information about input parameters to the user - those who execute the program - which makes it a horrible programming practice. One of much better ways is to pass input parameters in a given format, like 'length = 10' or 'fruit : apple'. Here's an example:

```
hojin@hojin-VirtualBox:~$ ./a.out length=10 density=5 area=240
```

One of the major benefits of this type of programming technique is that it does not require certain order to type in input parameters. If parameters were passed by values, and they were put in a different order, the program would give out a wrong result. By setting a format in command line inputs, the same value would be stored in the same variable name even if the order had been changed, and user conveninence would be enhanced. This programming practice requires a little more coding skills, but it does worth a cost.

2 How to split string inputs

The essential part of taking formatted command line input is to split the given string input. For example, if input is in the format of 'a=b', then we need to find and store 'a' and 'b' from the given input. How can we do this? There are at least three ways to do this

two C methods:

- loop through characters
- use sscanf

one of C++ methods:

- use std::string find, substr

C methods use char* instead of C++ style string. In this case, we can use *for* or *while* loop to search through the array of characters, and when '=' is found, we can store every previous letters into one variable and every later letters into another. That is the most intuitive way.

Another way is to use *sscanf*. *sscanf* converts char* variable into a given format, like int, float, double, or even char*. However, simply coding like this won't work:

```
sscanf(argv[i], "%s=%s", key, val);
```

This does not work because first char* identifier will take everything into variable 'key' and there will be nothing left for variable 'val'. The correct usage of sscanf will be like this:

```
sscanf(argv[i],"%[^=]%[=]%[^=]", key, jnk,val);
```

This code will put every character except for '=' into key, character '=' into jnk, and again every character except for '=' into val.

Another way to split string variables is to convert input into std::string type and use the inherent functions of the std::string container. Function std::string::find locates given character inside a string variable and returns the position of the character. Using find function, we can find where the character '=' is located. All that left is just to use std::string::substr function which parses string variable into specified positions of characters and extract key and values.

3 Few comments on project 6

- Do not use exit(1) for project 6. Throw all exceptions with CAAM420Exception class.
- lapack types conform to Fortran formats, so integers are in long type. Watch out when handling integers.

- In fortran, matrix is stored column-major. For example, matrix stored as (a b c d e f) in C would be stored as (a d b e c f) in Fortran. This way of storing matrix is also implemented in Clapack, so one has to be careful about how to call matrix elements.

- Final input parameter of dgesv, info is the error code. Various errors are represented with the value of info. One special case is when info gets positive number. It means given matrix is singular, and the linear system cannot be solved.