KIT205 Data Structures and Algorithms

Week 3 Tutorial

Creating a new project

We will be using Microsoft Visual Studio in this unit. You should follow the instructions below when creating a new project.

1. Start Microsoft Visual Studio.
2. Choose *New Project…* from the *File* menu.
3. Select the *Visual C++* template and choose *Win32 Console Application*.
4. Enter a name for the project (e.g. *Week3*) and check the location at the bottom of the *New Project* window (you probably want to put it on your network drive in a folder for KIT205). Click *OK*.
5. When the *Win32 Application Wizard* starts hit the *Next>* button.
6. Choose *Empty Project* and uncheck *Security Development Lifecycle (SDL)*, and click *Finish*.
7. From the *Project* menu, choose *Add New Item…*
8. Choose a new *C++ File* and give it a name (e.g. *week3.c*). Make sure that you change the filename extension from *.cpp* to *.c*.
9. The new empty file should now be open. Write a simple program (that is, a *main* method) that displays "hello world" on the console.
10. Run the program without debugging (*Ctrl F5*) so that the console window isn't immediately closed.

Code Analysis and Profiling

In this tutorial you will calculate the O() running time of the functions below, then implement them and run tests to see if the running times match your predictions.

11. Analyse (theoretically) the running time of the following functions:

```
int f1(long n){
    int k = 0;
    for (int i = 0; i < n; i++){
            for (int j = 0; j < n; j++){
                    k++;
            }
    }
    return k;
}

void f2(long n){
    int k = 0;
    for (int i = 0; i < n; i++){
            for (int j = 0; j < i; j++){
                    k++;
            }
            for (int t = 0; t < 10000; t++){
                    k++;
            }
```

```
    }
}
void f3(long n){
    if (n > 0){
            f3(n / 2);
            f3(n / 2);
    }
}

void f4(long n){
    if (n > 0){
            f4(n / 2);
            f4(n / 2);
            f2(n);
    }
}

void f5(long n){
    int k = 0;
    for (int i = 0; i < 10; i++){
            while (n > 0){
                    k++;
                    n = n / 2;
            }
    }
}

void f6(long n){
    f2(n);
    f3(n);
    f5(n);
}

void f7(long n){
    int k = 0;
    for (int i = 0; i < f1(n); i++){
            k++;
    }
    for (int j = 0; j < n; j++){
            k++;
    }
}
```

Now that we have theoretical times for the functions, let's check your predictions experimentally.

12. Copy your main function from last week (the one with a while loop for selecting options), but modify the while loop so that the options 1..7 select functions named f1..f7 (as defined above), calling them as *f1(n);* for example.
13. Next, add the time header to your project:
    ```
    #include <time.h>
    ```

14. In the while loop
    a. Create an *option* variable and scanf into *option* as before
    b. if *option* is not equal to 0,
        i. prompt for the value of *n* and set up a timer:

```
long n;
printf("Enter a value for n\n");
scanf("%d", &n);
clock_t start = clock();
```

        ii. Call the appropriate function, based on *option* and pass the value for *n*
        iii. Add the following code to print the time:

```
clock_t diff = clock() - start;
long msec = diff * 1000 / CLOCKS_PER_SEC;
printf("Operation took %d milliseconds\n\n", msec);
```

    c. else, if *option* is 0, set *quit* to 1 as before.
15. Now copy the functions into your project and record the times for various values of n in the spreadsheet provided. A good approach is to start with n=1 and then keep doubling the value of n until the program takes a few seconds to run.
16. Plot the data and confirm graphically that your O() predictions were valid. You may have to scale your O() functions as per the definition of O().

Note that I didn't include any exponential or factorial functions in this tutorial to save time, but rest assured that there are many simple functions (e.g. recursive Fibonacci) that are exponential, and problems where the *only* known solutions are exponential or factorial.