Lecture 1 : 코딩의 기본

자료구조는 컴퓨터과학에서 가장 기초적인 과목이며 Software 분야이든 Hardware 분야이든 컴퓨터과학에 관련된 모든 연구자는 반드시 이해하고 있어야 하는 내용을 담고 있다. 자료구조를 위한 언어에는 제한이 없지만, 이번 2021년도 강의에서 가장 보편적인 언어인 C++를 기준으로 설명한다. 동시에 C++에서 기본적으로 제공하고 있는 Standard Template Library를 기준으로 설명한다. 일반적으로 많은 대학에서 C++를 강의하고 있지만, STL 기반으로 강의하는 대학의 많지 않다. 일반적으로 많은 개발자들이 STL을 대학이 아니라 실제 작업 현장에서 많이 배우는 편인데 이렇게 ad hoc으로 배우기보다 체계적으로 학습하는 것이 더 바람직하다.

이 강의에서 제공하는 STL 지식은 실제 여러분이 갖추어야 할 내용의 25% 정도에 불과하므로 나머지는 스스로 익혀야 한다. 그리고 제공된 과제물을 해결할 때도 단순히 답을 낸다는 생각을 가지기보다 STL을 이용하여 더 간결하고 효율적으로 구성할 수 있도록 노력해야 한다. 본 수업을 위하여 매우 다양한 STL 실습 코드가 제공될 예정인데 수강생들은 이 코드를 반드시 수행해서 다양한 실험을 해야 한다.[1)]

1.0 다음 제시한 코딩 관련 표현이 어떻게 다른지 예를 들어 설명하시오.

Coder
Programmer
Chief-Programmer
Super-Programmer
System Designer
Hacker

1.1 자료구조를 전혀 모르면 어떤 일이 벌어질까요?

Sol) 자료구조를 잘 배운 사람과 그렇지 못한 사람은 어떻게 SW 개발을 하는지 비교하여 설명한다.

1.2. 정상적인 소프트웨어 개발 과정을 설명하시오.

- 학생 소프트웨어 (Student Software)의 대표적인 특징은 무엇일까요?
  이런 식으로 현장에서 작업을 진행하면 어떤 문제가 발생하는지 생각해보자.

우리의 목적은 “좋은” 자료구조를 만들고 이를 실전에서 활용하는 것이다. 그런데 문제는 어떤 것이 “좋은” 자료구조인가 하는 것이다. 일반적으로 “tree구조가 선형 배열보다 더 우수한 자료구조이다“라고 주장하는 사람들이 많은데 이는 올바르지 못한 표현이다. 우리가 좋은 자료구조를 만들기 위해서는 우리가 사용하고자 하는 동작(operation)을 먼저 정의해야 하고 이 동작을 최소화할 수 있는 자료구조가 해당 작업의 ”좋은 자료구조“가 되는 것이다.

---

1) 제공 코드를 고쳐서 다양한 확장 기능을 확인하고 그에 관하여 소스에 comment로 기록해야 한다.

즉 좋은 자료구조임을 판별하는 기준은 자료구조 자체의 구성요소, 구조가 아니라 그 자료구조에 어떤 동작을 할 것인가에 의해서 결정이 된다. 예를 들어 외부에서 어떤 원소가 입력되고 이들을 보관해야 하는 문제를 생각해보자. 만일 원소가 입력만 되고 추가의 작업, 예를 들어 탐색이나 삭제, 또는 같은 원소가 있는지(중복)를 체크(duplication check)가 없다면 이것은 일반적인 배열이면 충분하다. 예를 들어 STL vector를 사용해서 추가되는 원소를 해당 container에 넣어주기만 하면 된다. 그러나 만일 입력되는 원소가 이미 관리하는 원소와 같은지를 질문하는 동작이 자주 요청된다면 단순히 순서 없이 저장해두는 것은 매우 불편한 자료구조가 된다.[2)]

비유하자면 한 해에 1, 2번도 사용하지 않는 문(door)이라면 어떤 재료, 어떤 구조로 만들어도 상관이 없다. 그러나 그 문에 대하여 들어오고, 나가는 동작, 동시에 노크(knock) 등의 동작이 매우 빈번하게 일어난다면 문의 구조는 달라져야 한다. 만일 열리고 닫히는 동작이 드물다면 좌우로 여닫는 미닫이문이 유리할 것이고, 이 동작이 아주 잦다면 밀고 닫는 여닫이문이 더 유리하다. 대부분의 식당 입구 문을 앞뒤로 여닫는 여닫이문으로 만드는 이유는 여기에 있다. 이와 같이 “가장 좋은 문”이란 얼마나 많은 사람이 어떻게 이용하는가, 즉 그 문에 가해지는 동작의 특성에 달려있다. 여러분은 반드시 이 관점으로 자료구조에 접근해야 한다.

1.3. “좋은 software”란 어떤 software를 말하는지 자신의 기준으로 설명하시오.

1.4 Component Ware와 STL(Standard Template Library)의 의미는 무엇인가요?
- 소프트웨어 전공을 해서 돈을 버는 방법을 아는 대로 설명해 보시오.

1.5 프로그램 살펴보기, Test 하기, 분석하기, 검증(verify) 작업은 각각 어떻게 다른 것인가요?

1.6 Object-Oriented Programming과 일반 프로그래밍의 차이점을 설명하시오.
어떤 경우에 OOP 방법이 유리한지 설명하시오.

1.7 Coding Style에 대하여 아는 대로 설명하고, 자신만의 코딩 style을 10분 동안 공개적으로 설명하시오.

---

2) 2021년에 추가한 내용

1.8 다음 프로그램 코딩 방법에 대하여 자신의 의견을 말하시오.

- Visual Studio에서 int x[7000][7000]을 설정하는 것은 가능하다.
- int x[5000][5000]을 compile은 되지만 수행은 불가능하다.
- for loop을 10000번씩 이중으로 수행하는데 걸리는 시간은 약 2초 안쪽이다.
- int x[1], int x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[x[1]]]]]]]]]]]]]]]]]]]]]=10 ?
- Code Block에서 디버거를 사용할 수 있다.

1.9 과학과 공학에서 instrumentation이 무엇인지 설명하시오.

좋은 software의 가장 기본적인 속성은 "빠른" 수행시간(execution time)을 보장하는 것이며 이를 확인하기 위해서는 수행시간을 측정해야 한다. 수행시간이란 무엇인지 아래 용어를 기준으로 설명하시오.

- execution time
- CPU time
- turn-around time

```
// 특정 함수의 CPU 수행시간을 측정하는 함수
#include <iostream>
#include <chrono>
#include <cmath>
#include <functional>   // function을 변수로 넘길 수 있어요.
#include <vector>
#include <array>
using namespace std ;

#define N 10000000  // 천만번, 반드시 밖에 잡아야 한다.
array <long, N>  myarray ;
int   carray[N] ;

void C_style_insert(){
       for ( long i = 0; i < N; ++i )      {
              carray[i]=i ;
       }
}

void array_insert(){
       for ( long i = 0; i < N; ++i )      {
              myarray[i]=i ;
       }
}

void vector_insert(){
    vector <long >  myvector ;
       for ( long i = 0; i < N; ++i )      {
              myvector.push_back( i ) ;
       }
}
```

```
// 시간을 측정할 함수는 반드시 void fun( ) 로 만들어야 합니다.
//  잡다부리한 parameter 넣으면 안됩니데이....

double Time_Check( string MSG, function <void ( )> target ){
    chrono::system_clock::time_point mybegin, myend ;
    chrono::duration<double> sec ;
    double tsec ;

        mybegin = chrono::system_clock::now();
        target();  // 측정할 함수
        myend   = chrono::system_clock::now() ;
        sec = myend - mybegin ;

    tsec = sec.count() ;
    cout << "\n" << MSG << ": " << tsec << endl;
    return( tsec ) ;
} // end of Time_Check( )


int main(){

    Time_Check( "array insert",  array_insert) ;
    Time_Check("vector insert", vector_insert) ;
    Time_Check("c-스퇄 insert", C_style_insert) ;
        return 0;
}
```

1.10 여러분은 아래 제시된 프로그램 dogpig01.cpp 를 이용해서 다음을 측정해보시기 바랍니다. 단 하나의 명령어는 시간이 너무 짧아 측정이 불가능하므로 100만번 이상 그 수행시간을 누적 측정해보기 바랍니다.

```
int this, that ;
this = 890 ;
that = - 2378 ;

a) x = 1234 * 3678 ;
b) x = i + j ;
c) x = this - that ;
```

1.11 여러분이 설치한 컴파일러에서 아래 기본 연산의 시간을 측정하시오. 물리적인 시간과 integer add를 기본 단위 1t로 설정하여 이 기준으로 측정한 값도 제시하시오.

int add		float add		sin(x)	
int div int mult		float mult		abs(x)	
long add		double add		procedure call	
long div		double mult		your choice	

1.12 위의 문제에서 다음과 같이 중간 계산 코드를 사용하면 정확한, 또는 의미 있는 시간 측정이 왜 불가능한지 그 이유를 설명해 봅시다.

```
for(i=1 ; i < 10000 ; i++)
  for(j=1 ; j < 10000 ; j++)
     a) k = i + j ;
     b) k = i * j ;
     c) printf("\n k = %d", k) ;
```

1.13 **"좋은 코딩 습관은 평생의 재산이다"-**

a) local 변수와 global 변수
b) index 변수와 일반변수
c) temporary variable
d) 기타 전형적인 변수
- pointer, buffer , table , start, ending
e) pronounceable name

```
int ztypng[300]        ;  int num_CPU[10]
int zero_numbers[300]
int zjarh_dgkrtn[100][4]
int no_students_csed[100][4]
double whghksrb[ AKSTP]  ;
```

f) Do not use "real numbers" in practice.

int csed[150][5]

```
#define MAX_STUD  200
#define GRAde       4
#defne  EXTRA       2  // for 복학생
int csed[ MAX_STUD ][ GRADE + EXTRA ]
```

g) indentation 낭비의 예

나쁜 스타일	좋은 스타일 (Google 표준)

for(i=0; i < N; i++)
{
  x[i] = 0 ;
}
	for(i=0; i < N; i++){
	  x[i] = 0 ;
	}

for(i=0; i < N; i++) x[i] = 0 ;	for_set(x,1,N,0) ;

h) 부족한 어휘력

if( )
else if ( )
else if ( )
else if ( )
	switch(c) :
	  case '2'
	  case '3'
	  case '4'
	  case '5'

i) "줄을 서시오... "

o=x[i][j] n=x[i+1][j]+10 s=x[i][j+1] e=x[i-1][j]-31 w=x[i][j-1]+5	o = x[i][j] s = x[i][j+1] w = x[i][j-1] + 5 n = x[i+1][j] + 10 e = x[i-1][j] - 31

자료구조 과제물을 위한 프로그래밍 환경

Q) 왜 표준 C/C++를 사용해야 하는가 ?

a. Editing tool은 Linux 표준이 Code::Block을 사용해야 한다.

http://www.codeblocks.org/

b. 최신판을 설치하되 mingw-setup을 선택하여 compiler와 같이 설치해야 한다.

http://www.codeblocks.org/downloads

Windows XP / Vista / 7 / 8.x / 10:

File	Date	Download from
codeblocks-20.03-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net

c. MicroSoft의 Visual Studio는 표준 C/C++를 지원하지 않으므로 사용해서는 안된다.
만일 사용하게 되면 Portability가 없으므로 대부분 system에서 오류가 발생한다.

d. 다른 IDE(Integrated Development Environment)를 사용해도 좋다.

VSC (Visual Studio Code) - 최근에 가장 많이 사용하는 도구,
안되는 언어가 없다. but....

Eclipse (너무 무거움..)

Dev-C++ (너무 가벼움..)

VIM

notepad++