**Homework 2**

**2014313303 홍태하**

**1. Python 과 C (혹은 C++) 의 Data Type를 비교하고 효율성, 안정성, 확장성 면에서 장단점을 분석하시오.**

**Data type**

C언어의 data type에는 정수형, 실수형, 문자형 등이 있다. 정수형에는 short, int long 등의 data type이 있고 실수형에는 float, double, 문자형에는 char가 있다. 각각의 data type은 표현할 수 있는 data의 형태나 담을 수 있는 data의 크기가 다르다. 프로그래머는 data의 성질에 맞는 data type을 골라서 선언해야 한다.

Python의 data type에는 숫자형(정수, 실수, 허수, 8진수, 16진수)과 문자열, 배열 형태의 자료형으로 list, tuple, Dictionary가 있다. 숫자형과 문자열은 불가변한 자료형이고list와 Dictionary는 가변형이다. Tuple은 불가변형의 list이다. 여기서 가변형이라는 것은 A와 B가 같은 data를 가리키고 있을 때 B가 변하면 A도 같이 변하는 것이다. 불가변형이라는 것은 B가 변하더라도 A의 값은 유지되는 것을 말한다.

**C와 Python의 비교.**

Python에서는 C언어와 다르게 모든 변수가 극히 필요한 경우를 제외하고는 묵시적으로, 인터프리터에 의해 동적으로 바인딩 된다. 이 말은 사용자가 type을 지정해주지 않아도 인터프리터가 data의 type을 파악하여 내부적으로 변수에 자료형을 붙여준다는 것이다. 따라서 사용자가 일일이 생각해서 data type을 붙여줘야 하는 C언어와 다르게 python이 더 효율적이고 쉽게 프로그램을 작성할 수 있다.

Python은 또한 외부의 라이브러리로 쉽게 확장할 수 있고 새로운 데이터 타입으로도 이용할 수 있다. Python은 접착성이 뛰어나서 C언어와 같은 컴파일 언어나 다른 다양한 언어를 이용해서 가져올 수도 있고, Python을 가져다가 붙일 수도 있다. 따라서 확장성이 아주 좋다고 할 수 있다.

그러나 python이 아무리 효율적이고 확장성이 좋다고 하더라도 문자열 type의 변수를 숫자형으로 출력하려고 한다거나 변환하려고 하는 경우에는 자료형 관련한 에러가 발생한다. 그래서 안정성(reliability)부분에서 C언어가 더 장점이 많다고 할 수 있다.

**2. N 차원 array 의 A(i1, i2, ..., In) 의 주소를 결정하는 Accessing Formula 를 구하시오.**

: base address.

: Lower bound on subscript 1.

: Upper bound on subscript 1.

: Lower bound on subscript 2.

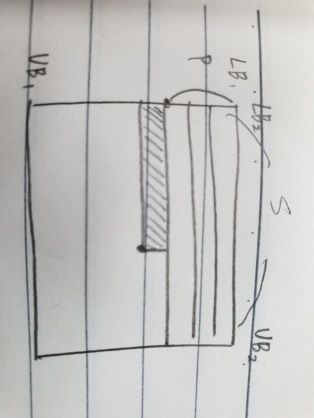
: Upper bound on subscript 2.

…

: Lower bound on subscript n.

: Upper bound on subscript n.

: Size of each component.

Row-major order에서 N차원 array의 주소를 결정하는 Accessing Formula 구하는 법.

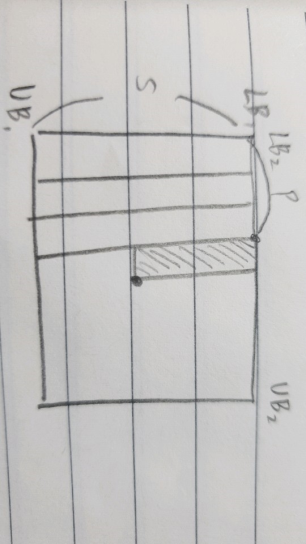
* n차에서 남게 되는 부분만 생각하면 (이다

example) 오른쪽의 2차원 array에서 S\*p를 제외한 색칠된 부분을 의미한다.

* n-1차에서 남게 되는 부분을 생각하면

가 된다.

* 이 과정을 1차까지 반복하면 밑의 식이 도출된다.



Column-major order에서 N차원 array의 주소를 결정하는 Accessing Formula 구하는 법.

* n차에서 남게 되는 부분만 생각하면

(이다

example) 오른쪽의 2차원 array에서 S\*p를 제외한 색칠된 부분을 의미한다.

* n-1차에서 남게 되는 부분을 생각하면

가 된다.

* 이 과정을 1차까지 반복하면 밑의 식이 도출된다.

**Row-major order**

(

**Column-major order**

(

**3. Pointer 를 사용할 때의 문제점 2개를 지적하고, 그 문제를 해결할 수 있는 방법을 제시하시오.**

**1) Garbage**

Garbage란 사용하지 않지만 메모리를 차지하고 있는 것을 말한다. 어떤 변수도 가리키지 않아 접근할 수도 없고 사용할 수 없는 공간을 의미한다.

문제점.

* Garbage가 점점 늘어나게 되면 쓸데없이 낭비되는 메모리 공간이 커지게 되고 프로그램에 할당해 줄 메모리의 양이 줄어든다. 심할 경우 프로그램이 멈출 수도 있다.

발생하는 경우.

* 동적 할당을 한 후 그 메모리를 가리키는 포인터를 다른 것을 가리키게 할 때 동적 할당이 된 메모리는 어떤 방법으로도 접근할 수 없게 되고 garbage가 된다.
* 동적 할당을 한 후 그 메모리를 회수하지 않았을 경우.

해결 방안.

* Garbage collection을 해서 사용 가능한 메모리의 양을 늘린다. .

**2) Dangling reference.**

Dangling pointer는 어떤 포인터가 해제되어 있는 메모리 영역을 가리키고 있을 때 생긴다. 즉, 객체에 대한 참조가 포인터 값에 대한 수정 없이 삭제되거나 할당 해제돼서 포인터가 계속 할당 해제된 메모리를 가리킬 때에 생긴다.

문제점.

* 메모리 접근 시 예측 불가능한 동작.
* 메모리 접근 불가 시 Segmentation fault.
* 보안 위험.

발생하는 경우

* 동적 할당 된 변수를 할당해제 할 경우.
* 지역변수를 가리키던 포인터를 return할 경우.

해결 방안.

* 모든 메모리를 할당 해제할 때 포인터의 초기화 보장(smart pointer).
* 메모리 할당 해제의 권한을 Garbage collector에 부여.
* 지역변수를 가리키던 포인터를 return할 때는 메모리를 동적 할당하여 사용하면 메모리가 Heap 영역에 저장되므로 의도한 결과값을 얻을 수 있다.