|  |  |
| --- | --- |
| 이름 | 신동욱 |
| 학번 | 20204101 |
| 학년 | 3 |
| 과목 | 프로그래밍언어론 |



프로그래밍언어론  
보고서 #1

Python, C, Java, C++ 4가지 언어에 대해 다양한 측면에서 평가하고 비교하는 것이 목적이다. 네 가지 언어에 대해 수업에서 강조된 핵심 평가 항목들인 판독성, 작성력, 신뢰성에 대해 각 언어들을 평가하고 분석할 것이다. 이러란 평가 및 비교를 통해 각 언어의 장단점을 이해하고, 언어의 특징을 잘 살려서 개발할 수 있는 개발자가 되고자 한다.

언어의 대표적인 평가 기준인 판독성, 작성력, 신뢰성은 각각 세부적인 특징을 가진다. 판독성은 단순성, 직교성, 데이터 타입, 구문 설계와 같은 특징들에 의해 결정된다. 작성력은 단순성과 직교성, 추상화의 지원, 표현력에 의해 결정된다. 신뢰성은 타입 검사, 예외 처리, 별칭, 판독성과 작성력에 의해 결정된다. 이하에서는 각 세부적인 특징들에 대해 Python, C, Java, C++ 4가지 언어에 대해 예시를 들고 평가하고자 한다. 다만, 모든 평가 항목에 대하여 언어의 모든 구조에 대해 평가하기에는 무리가 있기에, 특징에 대해 두드러지는 부분에 한하여 예시를 들고자 한다.

1. 판독성

판독성은 프로그램을 얼마나 쉽게 읽고 이해할 수 있는 지의 정도이다. 소프트웨어의 유지보수의 용이성은 이 판독성에 의해 결정된다.

판독성에 기여하는 특징은 단순성,

* 1. 단순성

언어의 단순성은 판독성에 상당한 영향을 미친다. 많은 기본 구조를 가진 언어는 더 적은 개수의 기본 구조를 가진 언어보다 배우기가 어렵다. 간단한 예시로, Java 언어에서의 public static void main(String []args) 가 그 예시이다. 이와 같은 메인 함수는 언어를 처음 배우는 입장에서 혼동을 불러올 수 있다. 반면 C와 C++는 int main() 와 같은 구조는 직관적이며 단순한 구조이다. Python은 더 나아가 메인 함수가 존재하지 않을 수 있다.

|  |  |
| --- | --- |
| count = 1 | Python |
| int count = 1 | C |
| int count = 1 | C++ |
| int count = 1 | Java |

위 4가지 언어에 대해 변수 선언과 동시에 할당 대해 다른 특징을 가진다.

C, C++, Java 에서는 변수 선언과 동시에 할당을 할 경우 변수에 대한 타입을 명시해야 한다 또한 변수 선언만 단독적으로 할 수 있다. 하지만 Python 에서는 변수를 선언할 시 타입을 명시하지 않아도 되고 동시에 값을 반드시 할당해야만 한다. 이는 4가지 언어 중 Python 만이 두드러 지는 특징이다.

두번째로 프로그래밍 언어는 **특징 다중성** 이라는 특징을 가진다. 이는 특정 연산을 수행하는데 한 가지 이상의 방법을 갖는 것이다.

count = count + 1

count += 1

count++

++count

C, C++, Java 는 정수에 1을 더하는 연산을 위와 같은 4가지 방법으로 수행할 수 있다. 반면 Python에서는 전위연산자, 후위연산자를 지원하지 않는다. 특징 다중성이 두드러지면 언어에 대한 단순성이 하락한다. C, C++, Java 는 Python보다 특징 다중성이 높다고 할 수 있기에, 단순성이 떨어진다고 할 수 있다.

세번째로 프로그래밍 언어는 **연산자 중복**이 문제가 될 수 있다. 연산자 중복은 대게 유용하다. 적절한 상황에서 연산자 중복을 사용한다면 단순성을 높일 수 있다. 그러나 사용자가 과도하게 생성하고 사용한다면 판독성을 감소시킨다.

C, Java 에서는 사용자가 연산자 중복을 제공하지 않는다. 반면 C++와 Python 에서는 연산자 중복을 제공한다. 필자의 개인적인 생각으로, 연산자 중복을 하는 것은 단순성이 올라갈 수는 있지만 판독이 어려울 수 있기에 오히려 혼동을 일으킬 수 있다고 생각한다. 가령 특정 객체 간 덧셈이 이루어질 때 ‘+’ 더하기 기호에 대해 중복을 정의하는 것이 아닌, 객체의 메소드로 add() 를 정의하는 것이 좀 더 좋은 방법이라고 생각한다.

단순성은

* 1. 직교성

직교성이란 상대적으로 적은 개수의 기본 구조들이 상대적으로 적은 개수의 방법으로 조합되어 언어의 제어 구조와 데이터 구조가 생성될 수 있다는 것을 의미한다. 가령 C++ 이나 Java 에서의 접근 제어가 그 예시가 될 수 있다. 접근 제어란 Public, Private 등 과 같은 외부에서 접근 가능한 범위를 지정하는 데에 사용되며, 이는 모든 변수에 대해 조합이 가능하다.

|  |  |
| --- | --- |
| **Python**  def create\_array():  arr = [1, 2, 3, 4, 5]  return arr | **Java**  int[] createArray() {  int[] arr = new int[5];  //배열을 채우는 코드  return arr;  } |
| **C++**  int\* createArray() {  int \*arr = (int\*)malloc(5 \* sizeof(int));  //배열을 채우는 코드  return arr;  } | **C**  int\* createArray() {  int\* arr = new int[5];  //배열을 채우는 코드  return arr;  } |

위 코드에서는 함수(또는 메소드) 에서 배열을 반환하고 있다. 가장 큰 차이점은 Python과 Java 에서는 함수 내에서 생성한 배열을 직접 반환할 수 있지만, C와 C++에서는 배열을 직접 반환할 수 없다. 그 대신 배열을 가리키는 포인터를 반환해야 한다. 이러한 구조는 Python과 Java는 배열의 반환 부분에서 직교성이 더 뛰어나다는 것을 확인할 수 있다.

* 1. 데이터 타입

데이터 타입과 구조를 충분히 정의하면 판독성이 향상한다.

|  |  |
| --- | --- |
| **Python**  x = True  if x:  … | **Java**  boolean is\_true = true;  if (is\_true) {  …  } |
| **C++**  bool is\_true = true;  if (is\_true) {  …  } | **C**  int is\_true = 1;  if (is\_true) {  …  } |

위 코드에서는 불리언(boolean) 타입을 사용하는 것을 볼 수 있다. Python, Java, C++는 불리언 타입을 제공하나, C에서는 제공하지 않는다. 하지만 정수 1과 0으로 불리언 타입을 대체하지만, 불리언 타입을 나타내는지, 혹은 정수 0과 1의 의미로 사용하는지에 대해서 판독하기 어렵다. 이는 다른 언어에 비해 판독성이 낮다고 볼 수 있다.

* 1. 구문 설계

언어의 구문은 프로그램의 판독성에 큰 영향을 미친다. 가령, 제어 구조에서 그룹을 형식화 하는 것, 즉 코드 블록을 나누는 방식은 매우 중요하다

|  |  |
| --- | --- |
| **Python**  if x > 0:  print("\n")  print("\n") | **Java**  if(x > 0) {  System.out.println("");  System.out.println("");  } |
| **C++**  if(x > 0) {  cout << "" << endl;  cout << "" << endl;  } | **C**  if(x > 0) {  printf("\n");  printf("\n");  } |

위 코드에서는 조건문에 대하여 조건문 안의 코드들이 어떤 식으로 블록을 나누는 지에 대해 보여준다. C 와 C++ Java 에서는 조건문 내부의 코드를 중괄호 { } 로 감싸는 것을 볼 수 있다. 반면 Python 에서는 들여쓰기로 블록을 구분한다.

1. 작성력

작성력은 선택된 문제 영역에 대해 프로그램을 생성하는 데 얼마나 쉽게 작성될 수 있는가에 대한 척도이다. 단순성과 직교성은 작성력에 지대한 영향을 미친다. 단순성은 코드를 이해하기 쉽고 유지보수하기 쉽게 만든다. 난이도가 줄어들기에 생산성을 향상시킨다. 직교성은 적은 수의 기본 구조들의 조합으로 일관된 규칙을 갖는 언어는 작성력을 증가시킨다.

* 1. 추상화의 지원

추상화는 많은 세부사항이 무시될 수 있는 방식으로 복잡한 데이터 구조나 연산을 정의하여 사용할 수 있는 능력을 의미한다.

프로세스 추상화는 복잡한 작업을 단순화 하는 것이다. 반복문, 제어문 등 이러한 구조가 없다면 코드는 매우 지저분하게 되고, 이로 인해 코드의 흐름과 목적이 저해된다.

데이터 추상화는 복잡한 데이터를 간결하고 명확한 형태로 단순화하는 것이다. 데이터의 세부 사항을 숨겨, 더 나은 간결성과 명확성을 제공하기에 이는 작성력이 증가하는 효과를 가지고 있다.

|  |  |
| --- | --- |
| **Python**  class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age | **Java**  public class Person {  private String name;  private int age;  …  } |
| **C++**  class Person {  public:  string name;  int age;  …  }; | **C**  typedef struct {  char \*name;  int age;  } Person; |

위 코드는 사람(Person)에 대하여 추상 데이터 타입을 정의한 것이다. C++, Java, Python은 객체지향 프로그래밍을 지원하기에 class와 같은 객체 형태로 지원하는 것을 볼 수 있다. 반면 C는 class 가 아닌 구조체로 이를 지원한다. 4가지 언어는 형태는 다르지만, 추상 데이터 타입을 각기 다른 방식으로 제공하는 것을 볼 수 있다.

* 1. 표현력

표현력은 언어가 계산을 명세하는데 있어서 상대적으로 편리한 방식을 제공하는 것이다. 연산자와 미리 정의된 함수의 지원 정도이다.

|  |  |
| --- | --- |
| **Java – for**  for (int i = 0; i < length; i++) {  System.out.print(arr[i] + " ");  } | **Java – for each**  for (int num : arr) {  System.out.print(num + " ");  } |
| **C++ - for**  for (int i = 0; i < length; i++) {  cout << arr[i] << " ";  } | **C++ - 범위 기반 for**  for (int num : arr) {  cout << num << " ";  } |

위 코드는 Java, C++ 에서의 각 배열을 순회하는 여러가지 방법을 정의한다. 기존의 for문으로 배열을 순회할 때는 시작 위치를 명세하고 배열의 크기를 기술하는 방식을 쓴다. 그러나 다른 for문(Java에서는 for-each, C++에서는 범위기반 for) 에서는 인덱스 변수를 관리할 필요가 없고 배열의 모든 요소를 순회할 수 있다. 이와 같이 셀 수 있는 요소(iterable)들을 모두 순회하기 위해서 다른 방식의 for문을 사용할 수 있으며 이는 C++와 Java 에서의 표현력이 높다고 할 수 있다.

C의 경우에서는 코드 예시로는 없지만, 원시적인 for 문만 제공하며, 셀 수 있는 요소를 전체 순회하는 다른 형태의 for문을 사용할 수 없다. 반면 Python은 오직 for – in 반복문만 사용할 수 있다. for – in 반복문도 위와 비슷하게 셀 수 있는 요소들을 순회할 수 있다. 또한 range(int n) 와 같이 인덱스 변수를 정의하고 관리할 수 있는 기능 또한 제공한다. 따라서 C는 원시적인 for만 제공하기에 표현력이 떨어진다. Python은 표현력이 떨어지지 않는다.

1. 신뢰성
   1. 타입 검사

신뢰성을 위해 컴파일 시나 실행 중에 타입 오류를 검사한다. 프로그램이 실행 중일 때에 타입 검사를 하는 것은 컴파일 시에 타입 검사를 하는 것 보다 비용이 비싸다. C와 C++, Java는 컴파일 시에 타입 검사를 한다. 이 언어들은 실행-시간 발생 가능한 타입 오류는 거의 일어나지 않는다. 반면 Python은 실행 도중 변수의 타입을 동적으로 결정한다. 이는 Python 만이 가진 장점이라고 할 수 있다. 하지만 이는 엄격한 타입 검사라고 할 수는 없다고 생각하기에, Python이 상대적으로 신뢰성이 떨어진다고 할 수 있다.

* 1. 예외 처리

프로그램이 실행-시간 오류를 올바르게 수정하고 계속 실행할 수 있게 처리하는 것을 예외 처리라고 한다. 이러한 기능은 코드 안정성을 높이고 신뢰성에 도움이 된다.

|  |  |
| --- | --- |
| **Python**  try:  …  except ExceptionType as e:  …  finally:  … | **Java**  try {  …  } catch (ExceptionType e) {  …  } finally {  …  } |
| **C++**  try {  …  } catch (ExceptionType e) {  …  } | **C**  예외 처리 기능을 제공하지 않는다 |

Python과 C++, Java 에서는 이러한 예외 처리를 지원한다. Python과 Java 의 예외 처리 구조는 거의 유사하지만, C++는 finally 절이 없는 것이 특징이다.

C언어는 예외 처리를 지원하지 않는다. 따라서 C언어는 다른 언어에 비해 신뢰성이 떨어진다.

* 1. 별칭

동일한 기억장소를 두 개 이상의 다른 이름으로 참조하는 것. 별칭은 프로그래밍 언어에서 위험한 특징이라고 인식되고 있다. 별칭은 신뢰성에 도움이 되지 않으므로 제한적으로 별칭을 사용해야 한다.

|  |  |
| --- | --- |
| **Python**  arr = [1, 2, 3]  alias = arr  alias[0] = 10 //arr[0]은 10 | **Java**  int[] arr = new int[]{1, 2, 3};  int[] alias = arr  alias[0] = 10; //arr[0]은 10 |
| **C++**  int a = 5;  int& r = a;  int& alias = r;  alias = 10; //a는 10 | **C**  int a = 5;  int \*p = &a;  int \*alias = p;  \*alias = 10; //a는 10 |

위 4개에 언어는 별칭을 다양한 방법으로 제공한다. 먼저 Python과 Java 에서 배열(리스트)을 생성하고, 다른 변수에 배열을 할당 후, 다른 변수의 인덱스로 배열의 값을 수정하면 기존의 배열의 값도 같이 수정된다. 배열은 연속된 메모리 공간에 값을 저장하기에, 일종의 제한된 포인터라고 볼 수 있기 때문이다.

C 에서는 포인터 타입을 제공한다. 포인터 타입을 통해 다른 변수의 메모리 주소를 저장하고 접근할 수 있다. 이는 C의 강력한 특징 중 하나이다. 하지만 어설프게 사용한다면 심각한 오류를 초래할 수 있다. 따라서 C는 신뢰성이 하락하는 요인이 되며, 숙달된 프로그래머가 사용해야 된다.

C++ 는 참조 타입을 지원한다. 이는 별칭과 매우 연관이 깊다. 참조 타입은 기존 변수에 대한 새로운 변수를 정의하는 것이기 때문에 별칭 그 자체라고 할 수 있다. C++는 C를 확장하여 만든 것 이기에, 위에서 설명한 C의 신뢰성 측면의 단점을 포함하고, 참조 타입도 제공하기에 신뢰성이 하락한다.

* 1. 판독성과 작성력

판독성과 작성력은 모두 신뢰성에 큰 영향을 미친다. 프로그램이 쉽게 작성되면, 올바를 가능성이 높다. 판독성은 유지보수의 단계에서 신뢰성에 영향을 미친다. 판독하기 어려운 프로그램은 작성하고 수정하기 어렵다.

1. 비용
   1. 프로그래머 교육 비용

프로그래머 교육 비용은 언어의 단순성과 직교성에 의해 결정된다. 이 보고서에서 평가한 단순성과 직교성에서는 Python이 두드러지는 모습을 보았다. 실제로 Python은 프로그래밍 입문자가 하기에 가장 좋은 언어인 것 같다.

* 1. 프로그램 작성 비용

프로그램 작성 비용은 언어의 작성력에 의해 결정된다. 이 보고서에서 평가한 작성력은 Python이 가장 좋은 모습을 보였는데, 사실 C를 제외한 세 언어는 현대에서 많이 쓰이는 언어인 만큼 작성력이 딸리지 않는다고 생각한다.

* 1. 프로그램 컴파일 비용

C와 C++는 컴파일 시 기계 코드로 직접 컴파일 된다. 반면 Java는 컴파일 시 Byte Code를 생성하고 이 Byte Code를 기계어로 변환하는 시간을 가지기에 상대적으로 C, C++에 비해 느리다고 볼 수 있다.

Python은 인터프리터 언어로 명시적인 컴파일 과정이 없다. 대신 실행할 때 동적으로 컴파일 된다. Python은 컴파일 시간에 비해 실행 시간이 더 길기에 컴파일 속도 자체는 짧다고 할 수 있다.

* 1. 프로그램 실행 비용

Java는 JVM이 완벽하게 로딩되어야 하기 때문에, JVM 로드에 시간이 필연적으로 들 수 밖에 없다. 하지만 로드 된 후 속도는 현대의 언어 답게 빠른 속도를 보여준다. 하지만 더욱 안정적이고 빠른 C, C++에 비하면 느리다고 한다. Python은 고급 언어로서 상대적으로 시간이 느리지만, 이 역시 현대 언어 답게 크게 느린 모습은 보이지 않는다.

* 1. 언어 구현 시스템의 비용

구현 시스템이 비싸거나 비싼 하드웨어에서만 실행된다면 이는 언어 자체가 널리 사용되지 않을 확률이 높다. 위 4가지 언어는 모두 오픈소스 이므로 현대에 와서 가장 인기가 높은 언어들이고, 이는 시스템의 비용이 낮다고 할 수 있다.

* 1. 신뢰성 부족에 따른 비용

C와 C++은 포인터를 통해 메모리에 직접 접근할 수 있는 것을 위에서 확인했다. 이 두 언어들은 높은 수준의 실력을 필요로 하고 이는 역시 신뢰성 부족에 따른 비용으로 이어진다고 할 수 있다.

* 1. 프로그램 유지보수 비용

4가지 언어에 대해서 유지보수 비용은 이 역시 현대에서 주로 사용되는 언어들이다. 지금까지 사용한다는 뜻은 프로그램 유지보수 비용이 훌륭하다는 뜻으로 해석할 수 있다.