**2024 Spring OOP Assignment Report**

과제 번호 : 3

학번 : 20230642

이름 : 이채영

Povis ID : chyng

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. **프로그램 개요**
   * 본 프로그램은 이름 궁합을 계산하는 프로그램이다.
   * 두 글자의 이니셜로 된 이름 두 개를 입력하면 서로의 이름이 교차로 적히고, 이웃한 값들끼리 계산을 반복해 최종적으로 이름 궁합이 출력된다. 이때 계산 중간과정도 모두 출력된다.
   * 별도의 data file은 필요하지 않으며, <iostream> 라이브러리만 사용되었다.
2. **프로그램의 구조 및 알고리즘**

* Class를 사용한 경우, 사용한 class에 대한 설명.
* 전체적인 알고리즘에 대한 설명.
* 각 변수들에 대한 설명. (for 문을 쓰기 위해서 등의 이유로 임시로 선언한 변수는 포함시키지 않음)
* 알고리즘
  + 문자형(char) 배열 name1, name2에 두 개의 이름이 각각 저장된다. cin을 통해 name1, name2를 입력받는다.
  + 새로운 문자형 배열 name\_row에 이니셜이 계산할 순서대로 배열의 첫번째 칸부터 마지막 칸까지 새롭게 저장된다.
  + name\_row의 이니셜들을 정수로 변환하고(아스키 코드에 의해) ‘%10’을 계산해 일의 자리 수만 남긴다. 이를 정수형 배열 num\_initial에 차례대로 저장된다.
  + num\_initial 배열에서 이웃한 수끼리 더하고 일의 자리만을 남긴다. (add 함수 이용) 이를 정수형 배열 num\_2nd에 저장한다.
  + num\_2nd 배열에서 이웃한 수끼리 더하고 일의 자리만을 남긴다. 이를 정수형 배열 num\_3rd에 저장한다.
  + 최종적인 이름 궁합을 정수형 변수 final\_answer에 저장하고, cout을 통해 출력한다. 이때 이름 궁합 계산은 percentage 함수를 이용한다.
* 사용자 정의 함수
  + add(int a, int b) : 배열에서 이웃한 수들을 더할 때 사용하는 함수이다. 배열에서 이웃한 수들이 a, b가 된다. (a+b)%10 연산을 수행해 두 수를 더한 후 일의 자리만 남긴다. 그리고 이 값을 리턴한다.
  + percentage(int a, int b) : 최종적으로 이름 궁합을 계산하는 함수이다. num\_3rd 배열의 첫 번째, 두 번째 수가 a, b가 된다. a, b는 독립적인 숫자였으므로 answer이라는 변수에 a\*10 + b 값을 저장해준다. Answer 값이 1~99의 정수라면 값을 그대로 리턴하고, 0이라면 100으로 바꾼 후 리턴하기 위해 연산을 수행해야 한다.   
      
    이때   
    answer += 100;  
    answer = (answer % 101) + (answer / 101)

의 연산을 수행할 수 있다. Answer이 0인 경우 두 번째 줄의 연산은 answer = 100 + 0 = 100이 된다. Answer이 1~99의 정수 x인 경우 두 번째 줄의 연산은 answer = {(x+100)%101} + {(x+100)/101} = (x-1) + 1 = x가 된다. 따라서 1~99의 정수는 값 그대로 리턴되고, 0은 100으로 바뀌어 리턴된다.

1. **토론 및 개선**
   * + 본 프로그래밍 과제를 수행하면서 c++의 기본적인 입출력, 변수의 명시적 형변환, 배열과 함수의 사용을 익힐 수 있었다. C++은 파이썬과는 다르게 아스키 코드를 이용해 문자형 변수를 정수형으로 변환하려면 명시적으로 표기를 해야 한다는 점을 깨달을 수 있었다.
     + branch문을 사용하지 않고 0은 100으로 바꾸고, 나머지 1~99의 정수는 그대로 나오도록 하는 함수를 구현하는 데 %, / operator의 성질을 활용하였다.
     + 반복문을 사용할 수 있다면 두 자리보다 더 긴 이름의 궁합도 계산할 수 있고, 더 효율적으로 중간 과정을 계산할 수 있을 것이다.
2. **참고 문헌**