“불 가능? 불가능!”(화재위험성평가 어플리케이션)

현대건설 기술교육원 Smart 안전 4 조

조장: 김민준

조원: 강혜나, 정세림, 이승연(보건)

Github address: https://github.com/serim0310/Repository_1/tree/main

1. 어플리케이션 목적

1)대상: 가연성 물질을 사용하는 모든 산업체

2)목적: 각 현장의 화재 위험 인자를 도출하여 사전에 화재위험성을 평가하고 화재를 예방함.

2. 어플리케이션의 네이밍의 의미

“불 가능? 불가능!” : “불 가능?” 은 화재 발생 가능성(possible)을 뜻하며, “불가능!”은 해당 앱을 사용하면 화재 발생이 불가능(impossible)해진다는 의미를 가지고 있음. 중의적 표현을 사용하여 사용자가 기억에 오래 남기 위해 재치있게 네이밍을 해보았다.

3. 어플리케이션의 활용 가치

- 화재를 예방하여 산업체의 경제적 손실을 막을 수 있음
- 화재 위험도를 데이터화하여 산업체 별 화재위험성을 평가, 비교 할 수 있음
  -> 화재 개선 대책 수립에 활용할 수 있음
- 위험도 기준을 초과 할 시, 경보 장치 알람이 울릴 수 있도록 설정하여 실제 화재 대피 훈련에 활용할 수 있음

4. 어플리케이션 개발 계획

-산업 현장에서 가장 많이 사용하는 10 대 가연성 물질, 점화원 수, 화재 하중, 온도, 습도를 고려하여 화재위험성을 사전평가하고 예방함.

- 평가 결과를 데이터화 하여 기록, 통계에 활용 할 수 있으며 추후 화재예방대책 수립에 데이터를 활용 할 수 있음.

1) 사용자로부터 인화점, 가연성, 화재 하중, 온도, 습도를 제공받는 함수 사용

2) 받은 데이터를 기반으로 화재 위험 점수를 공식대로 계산하는 함수 사용

3) 점수별 위험성을 분류하는 함수 사용

4) 데이터를 받아 엑셀에 저장하는 함수 사용

5) 사용자의 종료 전까지 반복되는 함수 사용

5. 개발 과정

A. 사업장에서 자주 사용하는 가연성 물질 데이터를 입력함 (이름, 인화점)

```
4  def classify_flammability(material):
5      # 물질별 인화점 정보 (물질: 인화점)
6      flash_points = {
7          "휘발유": -43,
8          "아세톤": -20,
9          "이황화탄소": -30,
```

B. 변수 설정: 가연성 물질 이름, 점화원 수, 화재하중, 온도, 습도

C. 가연성 물질별로 인화점을 입력하여 인화점에 따라 가연성 점수를 결정하고
물질정보가 없을 때는 “알 수 없는 물질입니다”로 입력함

```
# 인화점에 따른 가연성 점수를 결정합니다
if material in flash_points:
    flash_point = flash_points[material]
    if flash_point >= 60:
        return 1  # 낮은 가연성
    elif 30 <= flash_point < 60:
        return 2
    elif 0 <= flash_point < 30:
        return 3
    elif -20 <= flash_point < 0:
        return 4
    elif flash_point < -20:
        return 5  # 높은 가연성
else:
    return "알 수 없는 물질입니다."  # 물질 정보가 없을 때
```

D. 화재 위험 점수에 따라 화재 위험 등급을 매김

*화재 위험 등급 : < 20 (Very Low), <=40(Low), <=60(Moderate), <=80(High), <=100(Very high)

*화재 위험성 점수 계산

```python
def calculate_fire_risk(ignition_sources, fire_load, temperature, humidity, flammability):
    """
    화재 위험 점수를 계산합니다.
    ignition_sources: 점화원 수 (1-5)
    fire_load: 화재 하중 (1-5)
    temperature: 온도 (°C)
    humidity: 습도 (%)
    flammability: 가연성 (1-5)
    """
    # 온도와 습도의 영향을 고려하여 점수를 계산합니다
    temperature_factor = temperature / 30  # 온도의 영향을 30°C로 스케일링
    humidity_factor = (100 - humidity) / 50  # 습도가 낮을수록 위험 증가
    weather_factor = temperature_factor * humidity_factor

    return flammability * ignition_sources * fire_load * weather_factor  # 최종 화재 위험 점수
```

E. 값을 잘못 입력하였을 경우 프로그램이 종료되지 않고 “올바른 값을 입력해주세요”라는 명령어가 나옴

```python
def get_input(prompt, value_type=int):
    """
    사용자 입력을 받아 특정 타입의 값을 반환합니다.
    올바른 값이 입력될 때까지 반복합니다.
    """
    # 사용자로부터 입력을 받아 올바른 타입으로 변환합니다
    while True:
        try:
            value = value_type(input(prompt))
            return value
        except ValueError:
            print("올바른 값을 입력해주세요.")


def get_ranged_input(prompt, min_value, max_value):
    """
    사용자 입력을 받아 특정 범위 내의 정수를 반환합니다.
    """
    # 범위 내의 값을 입력받도록 합니다
    while True:
        value = get_input(prompt)
        if min_value <= value <= max_value:
            return value
        else:
            print(f"값은 {min_value}에서 {max_value} 사이여야 합니다. 다시 입력해주세요.")
```

F. 정상 값을 입력한 경우

```
화재 위험 평가 프로그램입니다.
[물질 보기] 휘발유, 아세톤, 이황화탄소, 등유, 니트로벤젠, 에탄올, 메탄올, 톨루엔, 헥산, 벤젠
물질 이름을 입력하세요 (종료하려면 '종료' 입력): 휘발유
점화원 수 (1-5): 5
화재 하중 (1-5): 4
온도 (°C): 25
습도 (%): 50
휘발유의 인화점 분류 점수는: 5점입니다.
화재 위험 점수: 83.33
화재 위험 등급: 매우 높음 (Very High)
```

E. 비정상값을 입력한 경우

```
화재 위험 평가 프로그램입니다.
[물질 보기] 휘발유, 아세톤, 이황화탄소, 등유, 니트로벤젠, 에탄올, 메탄올, 톨루엔, 헥산, 벤젠
물질 이름을 입력하세요 (종료하려면 '종료' 입력): 승연
알 수 없는 물질입니다.
```

6. 어플리케이션 개발 후기

화재사고 발생 시 중대 재해로 이어질 수 있는 위험성이 매우 크다. 사업장에서는 가연성 물질을 사용하고 있으며 화재사고의 점화원이 될 수 있는 장비를 빈번하게 사용하고 있어 화재사고 위험이 항상 상존한다. 화재 사고 예방을 위한 다양한 제도를 명시하고 있으나 작업 특성에 적합한 예방 제도를 적용하기에는 한계가 있다. 사업장에서 활용, 적용한다면 직접적인 원인의 화재사고 예방이 될 것으로 기대된다.

개발초기에는 사용자가 물질의 인화점을 입력하는 방식이었다. 사용자 입장에서 생각하여, 인화점을 입력하는 대신 가연성 물질 명칭을 넣도록 변경하였다. 최종적으로는 가연성 물질을 보기로 제시했다. 개발자는 의도한 바를 사용자가 불편하지 않게 구현해야 한다는 것이 중요하면서도 어려운 점인 것 같다.

어플리케이션의 경우 While 문을 사용하여 사용자가 종료하기 전까지 반복되어야 한다는 것을 놓치기 쉬운 것 같다. 교수님의 말씀으로 중간에 잘못 입력 했을 때 대비하는 코드는 작성했으나, 생길 수 있는 다른 에러들은 아직 예측이 불가하여 추후에 이런 점을 더 공부하고 싶었다.