화재 대피 시간 계산기

학번:2118042
학과:안전공학과
이름:권민준
https://github.com/minjjjun/HW01_-

계산기의 목적

이 계산기는 화재 상황에서 안전하게 대피하기 위해 필요한 최소 대피 시간을 계산합니다. 이는 건물의 총 인원 수와 출구의 수에 따라 결정됩니다.

계산기 개발 계획

1. 입력 변수
   - 건물의 총 인원 수
   - 건물의 총 출구 수

2. 연산 과정 설계

- 조건문: 사용자가 선택한 연산에 따라 해당 계산을 수행합니다.
- 반복문: 사용자가 프로그램을 종료할 때까지 반복하여 계산을 수행할 수 있도록 합니다.

계산기 개발 과정 및 후기

개발 과정

1. 구현 계획 수립: 사용자가 선택한 연산에 따라 해당 함수를 호출하여 결과를 반환하도록 설계하였습니다.

2. 코드 구현: Python으로 계산기를 구현하였습니다. 각 함수는 입력된 값에 따라 계산을 수행하고 결과를 반환합니다.

3. 디버깅 및 테스트: 각 함수를 개별적으로 테스트하고, 예외 처리를 추가하여 프로그램이 안정적으로 동작하도록 하였습니다.

에러 발생 지점 확인 및 해결 방법 제시

1. Division by zero 에러 처리: 사용자가 출구 수를 0으로 입력하는 경우, division by zero 에러를 방지하기 위해 예외 처리를 추가하였습니다.

```
def calculate_evacuation_time(total_population, total_exits):
    try:
        if total_exits == 0:
            return "Error: Total exits should not be zero."
        else:
            return total_population / total_exits
    except ZeroDivisionError:
        return "Error: Total exits should not be zero."
```

2. 잘못된 입력 처리: 사용자가 잘못된 값을 입력했을 때 에러를 방지하기 위해 예외 처리를 추가하였습니다.

```
try:
    total_population = int(input("Enter total population in the building: "))
    total_exits = int(input("Enter total number of exits: "))
except ValueError:
    print("Error: Please enter a valid number.")
```

해결 방법 적용 시 변화한 내용

- 에러 메시지가 더 명확해졌습니다.
- 프로그램의 사용자 친화성이 향상되었습니다.

개발 과정 캡쳐

```
def calculate_evacuation_time(total_population, total_exits):
    try:
        if total_exits == 0:
            return "Error: Total exits should not be zero."
        else:
            return total_population / total_exits
    except ZeroDivisionError:
        return "Error: Total exits should not be zero."

def main():
    while True:
        print("\nSafety Engineering Calculator - Evacuation Time Calculator")
        print("1. Calculate Minimum Evacuation Time")
        print("2. Exit")
        choice = input("Select an option (1/2): ")

        if choice == '1':
            try:
                total_population = int(input("Enter total population in the buildi
                total_exits = int(input("Enter total number of exits: "))

                evacuation_time = calculate_evacuation_time(total_population, total
                if isinstance(evacuation_time, str):
                    print(evacuation_time)
                else:
                    print(f"Minimum Evacuation Time: {evacuation_time} minutes")
            except ValueError:
                print("Error: Please enter a valid number.")

        elif choice == '2':
            print("Exiting the calculator. Goodbye!")
            break

        else:
            print("Invalid choice. Please select again.")

if __name__ == "__main__":
    main()
```

개발 후 느낀 점

이번 과제 주제를 화제 대피에 관한 주제로 선정한 이유는 안전공학 쪽으로 어떤 계산기가 우리 실생활에서 유용할지 고민해 보다가 실생활에서 제가 직접 경험했던 것을 통해 주제를 선정하면 좋지 않을까 생각이 들었습니다. 그래서 떠올리게 된 것은 현재 국원생활관에서 살고 있는데 모두가 다 자는 새벽 시간에 단순 오작동으로 인한 화재경보가 한 달 사이에 무려 세 번이나 발생하는 불상사가 일어났습니다. 시끄러운 사이렌 소리와 스트레스가 점점 올라가는 와중에 문득 의문이 들었습니다. 혹시 다음에도 똑같이 화재경보가 울린다면 그 때는 과연 오작동일까 진짜 화재가 일어났을 경우일까. 만약 정말 화재가 일어난 것이라면? 이런 반복되는 오작동, 자고 있던 와중에 난데없는 사이렌 소리와 스트레스 등에서 모두가 안전에 대한 불감증이 조금씩 생길 것이라고 생각합니다. 하지만 사이렌이 울린 순간에는 고민할 시간, 지체할 시간이 없다는 것을 뜻하고 긴박한 상황임을 누구나 바로 인지할 수 있어야 하는데 '이번에도 오작동일까'라는 생각이 드는 것이 위험할 수 있겠다라는 생각이 들었습니다. 실상황이라고 가정해보면 이미 그런 생각을 하고 있는 와중에도 불길이 빠르게 번지고 있기 때문입니다. 전방에서 군복무를 한 저는 이런 긴박한 상황, 사이렌 소리에 굉장히 예민할 것이라 생각했지만 그건 또 아니었던 것 같고 이런 반복되는 오작동 속에 가려지는 진짜 화재에 누군가는 울려대는 사이렌 소리를 무시하고 계속 잠을 잘 것을 생각하면 조금 무섭기도 합니다. 최악의 상황에는 그 잠에서 영영 깨지 못할 수도 있는데 말입니다. 만약 정말로 화재가 일어났을 경우에 탈출 시간이 지체되어 건물 내부에 고립되어 버리면 짧은 시간 내에 화재 연기, 즉 유독 가스로 인한 질식으로 이어지기 때문에 화재 현장, 건물에서의 신속한 탈출이 무엇보다 중요하다고 할 수 있습니다. 물론 주위에 소화기가 있고 초기 진압이 가능하겠다 싶은 경우에는 할 수 있지만 평범한 일상 속에서 화재를 예측, 또는 발견하기가 어려울 수 있습니다. 직접 화재를 겪어본 바로는 군복무 시절 흡연장에서 누가 불을 안 끄고 꽁초를 버렸는지 주위 낙엽들에 불꽃이 빠르게 번지는 것을 봤습니다. 물을 부어 초기 진압하려 했으나 효과가 미미했고 불길이 더 빠르게 번져서 재빨리 소화기로 진압을 했던 경험이 있습니다. 화재나 안전 사고 등은 언제, 어디서, 누구에게 일어날지 모릅니다. 항상 우리 모두가 화재, 안전에 대한 경각심을 갖고 평소 소화기 사용법이나 건물 내부 비상구, 화재 시 대피 동선 등을 꼭 숙지해야 한다고 생각합니다.

계산기의 효과

이 프로그램은 화재 대피 계획을 수립하고, 건물의 안전성을 평가하는 데 유용한 도구가 될 것입니다. 안전 엔지니어링에서 건물의 화재 대응 시스템을 설계하고 개선하는 데 큰 기여를 할 수 있습니다.