C++프로그래밍 및 실습

대학생 대상 금융 정보 제공 시스템 개발 프로젝트

최종 보고서

제출일자:2023.12.24
제출자명:김채흔
제출자학번:223445

1. 프로젝트 목표

1) 배경 및 필요성

처음 사회에 나오게 된 대학생들이 적금/예금 등의 경제활동을 시작할 때 정보를 알지 못해 신청하지 못하는 경우가 많음. 또한 지원해주는 정책을 알지 못해 혜택이 있지만, 이를 알지도 못해 신청하지 못하는 경우가 많음. 이를 방지하기 위해 각자 상황에 맞는 정책 정보나 금융 상품 등의 금융 정보를 제공해주는 프로그램이 필요함.

2) 프로젝트 목표

사용자의 상황에 따른 금융 및 정책 정보를 자동적으로 제공.

3) 차별점

기존 사이트는 예금, 적금 등 다양한 정보와 정책 관련 정보가 여기저기에 있어서 경제활동을 이제 처음 시작하는 대학생들이 정보에 접근하기에 쉽지 않음. 또한, 은행 별로 사이트가 전부 달라 하나하나 비교해보아야 함. 이를 보완하여 사용자의 상황에 맞는 금융 정보를 제공하여 대학생들이 손쉽게 정보를 얻을 수 있도록 함.

2. 기능 계획

1) 대학생 대상 정책 추천 기능

사용자가 입력한 상황에 따라 정책을 추천해 주는 기능

(1) 정부 지원 정책 데이터 세분화 및 사용자 입력 유효성 검증

현재 정부에서 지원하고 있는 청년 관련 정책들 정보를 수집하여 지역별,

분야별로 세분화함.

(2) 추천 시스템

지역/분야의 정보를 입력받고 조건에 해당하는 정책 정보를 출력

(3) 자세한 정보 제공

사용자가 정보를 자세히 알고자 하면 세부사항 출력(사이트 정보 포함)

2) 청년 대상 금융 정보 제공 기능

대학생들이 가입할 수 있는 청년 대상 금융 상품을 추천해준다.

(1) 은행 별 금융 상품 수집

은행 별 청년 대상 금융 상품을 수집한 뒤 카테고리 별로 세분화한다.

(2) 사용자 상황 맞춤 금융 정보 제공

은행, 지역 등 사용자의 상황을 입력받고 이에 해당하는 정보를 제공한다.

(3) 구체적인 정보 제공

사용자가 특정 정보에 대해 더 알고 싶어 할 경우, 관련 사이트 정보를 포함한 세부 정보들을 제공한다.

3. 기능 구현

(1) 정부 지원 정책 데이터 세분화 및 사용자 입력 유효성 검증

- 입출력

<financial.cpp>

- class YouthPolicy

Fields = 분야 카테고리 벡터

Regions = 정책 카테고리 벡터

- isValid함수

userinput = 사용자 입력 변수

ValidOptions = 유효성 검증할 벡터 변수

<csv.h>

- class CSVReader

string filename_ = CSV 파일의 경로를 저장하는 변수
char delimiter_ = CSV 파일에서 열을 구분하는 데 사용되는 구분자를 저장하는 변수
- CSVReader 생성자

filename = CSV 파일의 경로를 나타내는 문자열

char delimiter = CSV 파일에서 열을 구분하는 데 사용되는 구분자(',' 쉼표)
- readData함수

line = 파일에서 한 행을 저장하는 변수

data = CSV 파일의 데이터가 저장된 벡터

row = 한 행의 데이터가 저장된 벡터

- 설명

엑셀 데이터를 가져와 지역과 분야를 카테고리에 맞게 나눈 후, 분야와 정책 카테고리를 벡터로 설정해 사용자가 입력한 지역이나 분야가 카테고리에 있는지 검증한다.

- 적용된 배운 내용

클래스 : 정책 클래스를 만들어 정책 관련된 함수를 쉽게 쓸 수 있도록 함.

벡터 : 벡터를 활용하여 카테고리를 세분화

함수 : bool isValid(~) 함수를 활용하여 사용자 입력의 유효성 검증

클래스, 생성자 : 클래스와 생성자를 활용하여 financial.cpp에서 엑셀파일 데이터를 읽어옴

파일 입출력 : 엑셀 데이터를 불러와서 정보를 저장하여 출력

- 코드 스크린샷

<financial.cpp>

- isValid 함수

```
class YouthPolicy {
private:
    string FieldPolicy;//분야 입력 받는 변수
    string RegionPolicy;//시역 입력 받는 변수
    string input;
    //분야 벡터
    vector<string> Fields = { "일자리", "주거", "교육", "복지", "신여" };
    //시역 벡터
    vector<string> Regions = { "중앙부처", "서울", "부산", "대구", "인신", "광주", "대전", "울신", "경기", "강원", "충북"
   // 필터링 된 정책 벡터
    vector <string> Policy = {};

//TO DO: 세부 기능 2 : 사용자가 입력한 조건에 맞는 정책 출력
public:
    //유효성 검증 함수
    bool isValid(const string& userInput, const vector<string>& validOptions) {
        return userInput == "0" || find(validOptions.begin(), validOptions.end(), userInput) != validOptions.end();
    }
```

<csv.h>

-CSVReader. readData 함수

```
#include <sstream>
#include <vector>
#include <string>
int num; //조건에 맞는 상제 번호 변수
using namespace std;
//엑셀 읽어오는 클래스
class CSVReader {
public:
    CSVReader(const string& filename, char delimiter = ',') : filename_(filename), delimiter_(delimiter) {}
    //엑셀 파일 읽어오기
    vector<vector<string>> readData() {
        ifstream in(filename_);
        if (!in.is_open()) {
            cerr << "Failed to open file: " << filename_ << endl;
            return {};
        }

        vector<vector<string>> data;
        string line;
        string currentRow;
        while (getline(in, line)) {
            currentRow += line;

            // 큰 텍스트 블록을 묶은 경우 처리
            if (isInsideQuotes(currentRow)) {
                continue;
            }
```

```
        // 큰 텍스트 블록을 묶은 경우 처리
        if (isInsideQuotes(currentRow)) {
            continue;
        }

        vector<string> row = readRow(currentRow);
        if (!row.empty()) {
            data.push_back(row);
        }

        // currentRow 초기화
        currentRow.clear();
    }

    in.close();
    return data;
```

(2) 추천 시스템

- 입출력

<financial.cpp>

-Field, Region, PolicyShow 함수

FieldPolicy = 사용자에게 분야 입력받는 변수

RegionPolicy = 사용자에게 지역 입력받는 변수

<csv.h>

- PrintValue 함수

columnName1 = 첫 번째 열의 열 이름

value1 = 첫 번째 열에서 찾고자 하는 값

columnName2 = 검색하려는 두 번째 열의 열 이름
value2 = 두 번째 열에서 찾고자 하는 값
outputColumn = 출력하려는 열의 열 이름
data = CSV 파일에서 읽은 전체 데이터 벡터
columnIndex1, columnIndex2 = 각각 첫 번째 및 두 번째 검색 열의 인덱스
outputIndex = 출력하려는 열의 인덱스

- 설명

사용자가 분야와 지역을 입력하면 그에 해당하는 행을 찾고, 해당 행에서 정책 정보 열의 값을 출력한다.

만약, 사용자가 입력한 조건에 맞는 정책이 없다면 메세지를 출력한다.

- 적용된 배운 내용

반복문(while) : 사용자가 올바른 입력을 할 때까지 오류 출력

반복문(for) : 사용자가 입력한 값에 해당하는 행 찾고 출력하는데 활용

함수 : 사용자에게 분야와 지역을 입력받는 코드 void Field(), void Region()로

함수화하여 활용

: void PolicyShow()로 사용자가 입력한 조건에 맞는 정책 출력

헤더 파일, 함수 : 엑셀 데이터에서 사용자가 입력한 조건에 맞는 정책을 가져오는 코드를 csv.h 의 PrintValue() 함수로 사용

- 코드 스크린샷

<financial.cpp>

- Field 함수

```
//TO DO: 세부 기능 2 : 사용자가 입력한 조건에 맞는 정책 출력
public:
    //유효성 검증 함수
    bool isValid(const string& userInput, const vector<string>& validOptions) {
        return userInput == "0" || find(validOptions.begin(), validOptions.end(), userInput) != validOptions.end();
    }

    // 청년 정책 분야 입력 함수
    void Field() {
            cout << "정책 분야를 선택하세요(종료 시 0)" << endl;
            cout << "--------------------------------" << endl;
            cout << "일자리/ 주거/ 교육/ 복지/ 참여" << endl;
            cout << "--------------------------------" << endl;
            cin >> FieldPolicy;

            while (!isValid(FieldPolicy, Fields)) {
                cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
                cin >> FieldPolicy;
            }
    }
```

- Region 함수

```
// 청년 정책 지역 입력 함수
void Region() {
    cout << "지역을 선택하세요(종료 시 0)" << endl;
    cout << "-------------------------------------------------------------------------------------------" << endl;
    cout << "중앙부처/서울/부산/대구/인천/광주/대전/울산/경기/강원/충북/충남/전북/전남/경북/경남/제주/세종" << endl;
    cout << "-------------------------------------------------------------------------------------------" << endl;
    cin >> RegionPolicy;

    while (!isValid(RegionPolicy, Regions)) {
        cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
        cin >> RegionPolicy;
    }
}
```

<csv.h>

-PrintValue 함수

```
//사용사가 입력힌 조건에 맞는 정책 출력 함수
void PrintValue(const string& columnName1, const string& value1,
    const string& columnName2, const string& value2,
    const string& outputColumn) {
    int num   1; // 정책 번호 초기화
    vector<vector<string>> data = readData();
    //데이터 유효성 검증
    if (data.empty()) {
        cerr << "Empty data" << endl;
        return;
    }

    // 입력에 해당하는 열 찾기
    int columnIndex1 = -1;
    int columnIndex2 =  1;
    for (size_t i = 0; i < data[0].size(); ++i) {
        if (data[0][i] == columnName1) {
            columnIndex1 = static_cast<int>(i);
        }
        if (data[0][i] == columnName2) {
            columnIndex2 = static_cast<int>(i);
        }
    }

    if (columnIndex1 == -1 || columnIndex2 == -1) {
        cerr << "One or both columns not found" << endl;
        return;
    }
```

```
    // 사용자가 입력한 값에 맞는 리스트 출력
    bool found   false; // 조건에 맞는 금융 상품 유효성 검증
    for (size_t i   1; i < data.size(); ++i) {
        if (columnIndex1 < data[i].size() && columnIndex2 < data[i].size() &&
            data[i][columnIndex1]    value1 && data[i][columnIndex2]    value2) {
            found = true;
            if (outputColumn == "") {
                for (const auto& cell : data[i]) {
                    cout << " [" << cell << "]" << "₩t";
                }
            }
            else {
                int outputIndex =  1;
                for (size_t j = 0; j < data[0].size(); ++j) {
                    if (data[0][j] == outputColumn) {
                        outputIndex = static_cast<int>(j);
                        break;
                    }
                }

                if (outputIndex != -1) {
                    cout << num << "[" << data[i][outputIndex] << "]";
                    num += 1;
                }
                else {
                    cerr << "Output column not found" << endl;
                }
            }
```

```
    }
    // 해당하는 상품이 없을 경우 메시지 출력
    if (!found) {
        cout << "해당하는 상품이 없습니다." << endl;
    }
}
```

(3) 자세한 정책 정보 제공

- 입출력

<financial.cpp>

FieldPolicy = 사용자에게 분야 입력받는 변수

RegionPolicy = 사용자에게 지역 입력받는 변수

PrintInfo() = 특정 정책 자세한 정보 출력 함수

<csv.h>

- getInfo함수

Region: 검색하려는 지역
Field: 검색하려는 분야
filename_ = CSV 파일의 경로를 저장하는 변수

file = 파일을 열 때 사용되는 변수

line = 한 행의 데이터를 저장하는 문자열

iss = 한 행의 csv 데이터를 , 로 구분하기 위한 변수

token = 구분한 각 열 값 저장 변수

tokens = 열 값을 모으기 위한 벡터

result = 조건 만족하는 행을 저장 벡터

- PrintInfo 함수

info = 조건 만족하는 행들 저장 벡터

row = info의 행을 하나씩 읽기 위한 변수

cell = 행에서 열 값을 하나씩 읽기 위한 변수

-isnumber 함수

str = 사용자가 입력한 번호

number = 정수로 변환될 값

- 설명

사용자가 입력한 정책 번호에 해당하는 정책의 세부 정보를 모두 출력한다.

사용자가 입력한 번호가 숫자가 아닐경우, 숫자를 입력하도록 반복하고 사용자가 입력한 번호의 정책이 없을 경우 메세지를 출력한다.

- 적용된 배운 내용

조건문 : 사용자가 입력한 번호가 정수인지 판별하고, 정수라면 true 반환

: 사용자가 입력한 번호가 유효한지 검증

: csv.h 에서 파일이 열리지 않았을 때 오류 출력

: 데이터의 각 열에 맞는 정보 분류(지역, 정책명 등) 출력

: 사용자에게 0을 입력받으면 종료

헤더 파일: 정책의 세부정보를 출력하는 코드를 헤더파일 csv의 PrintInfo 함수를 호출하여 활용

벡터 : 특정 조건 행을 찾고 이를 반환하는 함수에서 여러 변수를 벡터로 설정

: 필터링 된 정책을 벡터로 저장

반복문(while) : 데이터의 행과 열을 읽어 저장할 때 활용

함수 : PrintInfo, PolicyShow 함수를 활용하여 자세한 정보를 출력할때 필요한 코드를 함수화하여 사용

- 코드 스크린샷

<financial.cpp>

```
    csvReader.PrintInfo(RegionPolicy, FieldPolicy);
}
```

<csv.h>

```
//입력이 정수인지 판별하는 함수
bool isnumber(const std::string& str) {
    istringstream iss(str);
    int number;
    iss >> noskipws >> number;
    return iss.eof() && !iss.fail();
}
```

- isnumber 함수

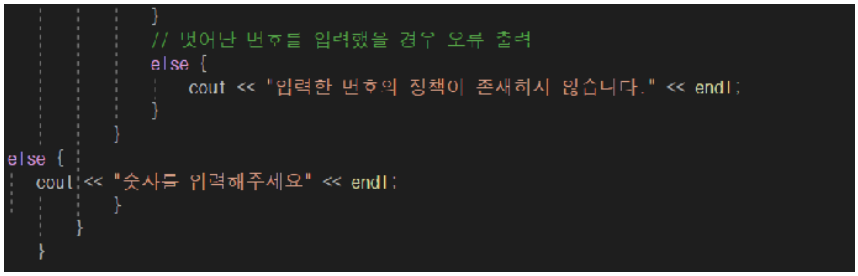
-getInfo 함수

```
// CSV 파일에서 특정 조건을 만족하는 행을 저장하는 함수
vector<vector<string>> getInfo(const string& Region, const string& Field,const int&col1,const int&col2) {
    vector<vector<string>> result;

    ifstream file(filename_);
    if (!file.is_open()) {
        cerr << "파일을 열 수 없습니다." << endl;
        return result;
    }

    std::string line;
    string currentRow;
    bool insideQuotes = false;
    while (getline(file, line)) {
        currentRow += line;
        // 큰 텍스트 블록을 묶음 예외 처리
        insideQuotes = isInsideQuotes(currentRow);

        // 큰 텍스트 블록이 끝났을 때
        if (!insideQuotes) {
            vector<string> tokens;
            istringstream iss(currentRow);
            string token;

            while (getline(iss, token, ',')) {
                tokens.push_back(token);
            }
```

```
            // 특정 조건을 만족하는 행인지 확인
            if (tokens.size() >= col2+1 && tokens[col1] == Region && tokens[col2] == Field) {
                result.push_back(tokens);
            }

            // currentRow 초기화
            currentRow.clear();
        }
    }

    file.close();
    return result;
```

-PrintInfo 함수

```
void PrintInfo(const string& Region, const string& Field) {
    vector<vector<string>> info = getInfo(Region, Field,2,32);
    int input = 0;
    // 저장된 행들 출력
    while (true) {
        cout << "자세한 정보를 보고 싶은 업체의 번호를 입력하세요(없으면 0)" << endl;

        string inputStr;
        cin >> inputStr;

        if (inputStr == "0") {
            break; // 0이 입력되면 종료
        }

        if (isnumber(inputStr) == true) {
            // stoi를 사용하여 문자열을 정수로 변환
            input = stoi(inputStr);

            if (input > 1 && input < info.size()) {

                // 해당하는 행의 열들 출력
                for (const auto& cell : info[input - 1]) {
                    // 열에서 필요한 정보만 출력
                    if (cell != "") {
                        if ((&cell == &info[input - 1][0] || &cell == &info[input - 1][0] + 9) or (&cell == &info[input - 1][0] + 32))
                            switch (&cell == &info[input - 1][0]) {
                            case 2:
                                cout << "지역: [" << cell << "]" << endl;
                                break;
```



(4) 은행 별 금융 상품 수집

- 입출력

<financial.cpp>

class YouthFinancial = 금융정보 관련 기능 클래스

FinancialType = 사용자에게 금융상품 종류 입력 받는 변수

BankType = 사용자에게 은행명 입력받는 변수

ManualInput= 설명 원하는 용어 입력받는 변수

FinancialItem = 금융상품 종류 벡터 변수

Banks = 은행명 변수

Deposit = 예금 종류 변수

saving = 적금 종류 변수

interest = 금리 계산 방법 변수

Finan = 금융 용어 변수

- isValid 함수

userInput = 유효성 체크할 사용자 입력 변수

ValidOptions = 유효성 체크할 벡터 변수

<csv.h>

CSVReader = 엑셀 데이터 불러오는 클래스(앞선 정책클래스에서 활용한 것과 동일)

- 설명

사용자에게 입력받은 은행명과 금융 상품 종류를 세분화하고, 사용자가 해당 카테고리 안에서 입력했는 지 유효성 검증을 한다.

- 적용된 배운 내용

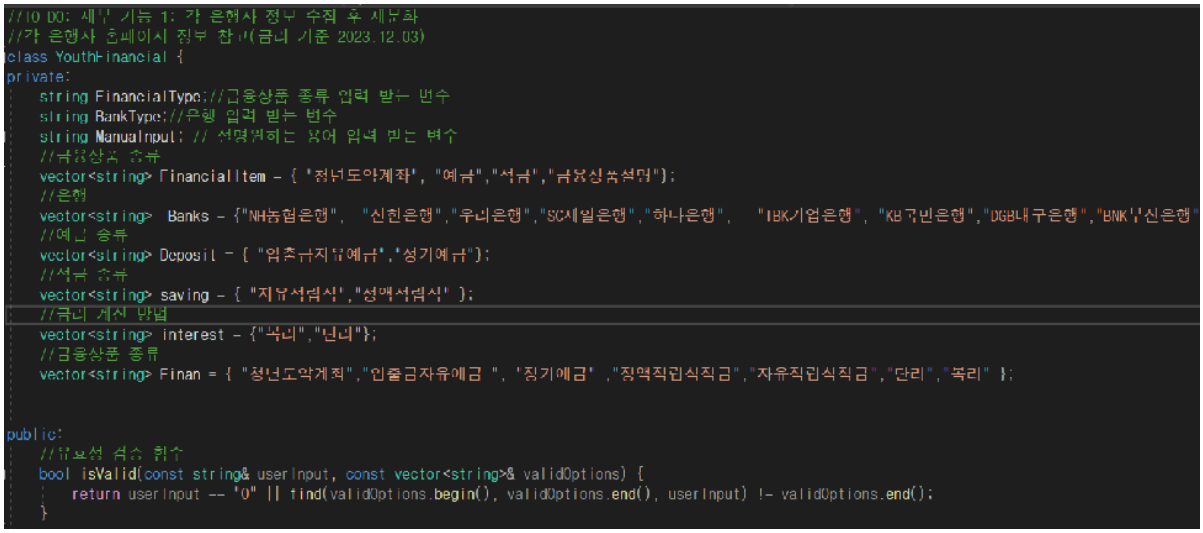
함수 : 유효성 검증을 함수화하여 지속적으로 사용할 수 있도록 함

벡터 : 은행 명과 금융 상품 종류를 벡터로 설정해 후에 기능 구현에 유용하도록 함

- 코드 스크린샷

<financial.cpp>

- isValid 함수



```
//TO DO: 세부 기능 1: 각 은행사 정부 수정 후 세분화
//각 은행사 홈페이지 정부 참고(금리 기준 2023.12.03)
class YouthFinancial {
private:
    string FinancialType;//금융상품 종류 입력 받는 변수
    string BankType;//은행 입력 받는 변수
    string ManualInput; // 설명하는 동안 입력 받는 변수
    //금융상품 종류
    vector<string> FinancialItem = { "청년도약계좌", "예금","적금","금융상품설명");
    //은행
    vector<string>  Banks = {"NH농협은행",  "신한은행","우리은행","SC제일은행","하나은행",   "IBK기업은행", "KB국민은행","DGB대구은행","BNK부산은행"
    //예금 종류
    vector<string> Deposit = { "임촉규지당예금",'정기예금"};
    //적금 종류
    vector<string> saving = { "자유적립식","정액적립식" };
    //금리 계산 방법
    vector<string> interest = {"복리","단리"};
    //금융상품 종류
    vector<string> Finan = { "청년도약계좌","인출금자유예금 ", '정기예금' ,"정액적립식적금","자유적립식적금","단리","복리" };


public:
    //유효성 검증 함수
    bool isValid(const string& userInput, const vector<string>& validOptions) {
        return userInput == "0" || find(validOptions.begin(), validOptions.end(), userInput) != validOptions.end();
    }
```

(5) 사용자 상황 맞춤 금융 정보 제공

<financial.cpp>

- Bank 함수

BankType = 사용자 입력 은행명

- Item 함수

FinancialType = 사용자 입력 금융상품 정보

- inputdeposit 함수

userdeposit = 사용자 입력 예금 종류

- Saving 함수

userdeposit = 사용자 입력 적금 종류

-ManualShow 함수

ManualInput = 사용자가 보고자 하는 금융상품 입력 변수

<csv.h>

- GetInfo함수

bank = 출력하려는 은행명
filename_ = CSV 파일의 경로를 저장하는 변수

file = 파일을 열 때 사용되는 변수

line = 한 행의 데이터를 저장하는 문자열

iss = 한 행의 csv 데이터를 , 로 구분하기 위한 변수

token = 구분한 각 열 값 저장 변수

tokens = 열 값을 모으기 위한 벡터

result = 조건 만족하는 행을 저장 벡터

- PrintValue 함수 (정책 정보 출력과 동일한 함수 사용)

- 설명

사용자가 금융 상품 종류와 은행 명을 입력하면 그에 맞는 금융상품들을 출력한다.

예적금의 경우, 종류, 금리 계산 방법을 출력하고 사용자의 입력에 따른 금융상품들을 출력한다.

사용자의 입력이 카테고리 안에서 벗어나지 않았는 지 판별하고, 벗어났다면 메세지를 출력하여 카테고리 안에서 입력할 수 있도록 한다.

- 적용된 배운 내용

반복문 : 사용자에게서 입력받을 때 유효성 검증

: 파일 유효성 검증

파일 입출력 : 각 금융상품에 맞는 데이터를 가져와 출력

- 코드 스크린샷

<financial.cpp>

-Bank 함수

```
//10 DO: 세부 기능 2 : 사용자 상황 맞춤 금융 정보 출력
// 은행별 위젯 함수
void Bank() {
    cout << "은행을 선택하세요" << endl;
    cout << "-----------------------------------------------------------------
    cout << "NH농협은행 /신한은행 /우리은행 /SC제일은행 /하나은행 /IBK기업은행
    cout << "-----------------------------------------------------------------
    cin >> BankType;

    while (!isValid(BankType, Banks)) {
        cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
        cin >> BankType;
    }
}
```

-Item 함수

```
// 금융 상품 입력 함수
void Item() {
    cout << "금융 상품을 선택하세요(종료 시 0)" << endl;
    cout << "---------------------------------" << endl;
    cout << "청년도약계좌/예금/적금/금융상품설명" << endl;
    cout << "---------------------------------"<< endl;
    cin >> FinancialType;

    while (!isValid(FinancialType, FinancialItem)) {
        cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
        cin >> FinancialType;
    }
}
```

-Inputdeposit 함수

```
//예식금, 대출 상보는 은행연합회 상보 사용(금리 기준 2023.12.20)
//사용사에게 예금 종류 입력받기
string userdeposit; //사용사 입력 예금 종류
void inputdeposit() {
    cout << "알고싶은 예금 종류를 선택하세요(종료 시 0)" << endl;
    cout << "                                    " << endl;
    cout << "입출금자유예금/ 정기예금" << endl;
    cin >> userdeposit;

    //사용자 입력 유효성 검증
    while (!isValid(userdeposit, Deposit)) {
        cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
        cin >> userdeposit;
    }
}
```

-Saving 함수

```
//사용자에게 적금 종류 입력받기
void Saving() {
    cout << "알고싶은 적금 종류를 선택하세요(종료 시 0)" << endl;
    cout << "---------------------------------------" << endl;
    cout << "정액적립식/ 자유적립식" << endl;
    cin >> userdeposit;

    //사용자 입력 유효성 검증
    while (!isValid(userdeposit, saving)) {
        cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
        cin >> userdeposit;
    }
}
```

-ManualShow 함수

```
void ManualShow() {
    cout << "설명을 원하는 금융 용어를 선택해주세요." << endl;
    cout << "-------------------------------------------------------------------------------" << endl;
    cout << "청년도약계좌/입출금자유예금/정기예금/정액적립식적금/ 자유적립식적금/ 단리/복리" << endl;
    cout << "-------------------------------------------------------------------------------" << endl;
    cin >> ManualInput;
    //사용자 입력 유효성 검증
    while (!isValid(ManualInput, Finan)) {
        cout << "잘못된 입력입니다. 다시 입력하세요." << endl;
        cin >> ManualInput;
    }
```

<csv.h>

-GetInfo 함수

```
//특정 은행 정보 희망계좌 정보 가져오기
vector<vector<string>> GetInfo(const string& bank) {
    vector<vector<string>> result;

    ifstream file(filename_);
    if (!file.is_open()) {
        cerr << "파일을 열 수 없습니다." << endl;
        return result;
    }

    std::string line;
    string currentRow;
    bool issideQuotes = false;
    while (getline(file, line)) {
        currentRow += line;
        // 큰 텍스트 블록을 묶은 경우 처리
        issideQuotes = isInsideQuotes(currentRow);

        // 큰 텍스트 블록이 끝났을 때
        if (!issideQuotes) {
            vector<string> tokens;
            istringstream iss(currentRow);
            string token;
```

```
            while (getline(iss, token, ',')) {
                tokens.push_back(token);
            }

            // 특정 조건을 만족하는 행인지 확인
            if (tokens[0] == bank) {
                result.push_back(tokens);
            }

            // currentRow 초기화
            currentRow.clear();
        }
    }

    file.close();
    return result;
}
```

(6) 구체적인 금융 상품 정보 제공

<financial.cpp>

- deposit 함수

userinterest = 사용자 입력 금리 계산 방법 변수

userdeposit = 사용자 입력 금융 상품 종류 변수

FreeDeposit = 은행 별 예금, 정액적립식 적금 자세한 정보 출력 함수

FreeSaving = 은행 별 자유적립식 적금 자세한 정보 출력 함수

- FinancialShow 함수

BankType = 사용자 입력 은행 종류 변수

YouthAccount = 은행 별 청년도약계좌 자세한 정보 출력 함수

- ManualShow 함수

ManualInput = 사용자가 설명 원하는 금융 용어 변수

FinancialManual = 입력 받은 금융 용어 자세한 설명 출력 함수

<csv.h>

- YouthAccount 함수

info = 입력받은 은행의 청년도약계좌 데이터 저장

cell = 데이터 각 열의 문자열 값

- FreeDeposit,FreeSaving 함수

bank = 사용자에게 입력받을 은행 종류

interest = 사용자에게 입력받은 금리 계산 방법 종류

info = 입력받은 은행(금리 계산 방법)의 예적금 데이터 저장

inputStr = 사용자에게 입력받은 값

input = 사용자에게 입력받은 정책 번호

cell = 데이터 각 열의 문자열 값

- FinancialManual 함수

name = 사용자에게 입력받은 금융 용어 변수

info = 금융 용어 설명 데이터 저장

cell = 데이터의 각 열의 문자열 값

- 적용된 배운 내용

조건문 : 사용자의 입력에 따른 각 금융상품 데이터를 출력할 때 사용

: 사용자의 입력 유효성 검증

클래스, 함수 : 헤더파일에서 작성한 함수를 사용하고, 값을 입력받아 비슷한 방식으로 처리할 때에 이미 만들어진 함수를 가져와 사용

반복문 : swith문을 활용하여 데이터의 정보를 사용자가 보기 쉽게 출력

- 코드 스크린샷

<financial.cpp>

```
csvReader.PrintValue("은행", BankType, "은행", BankType, "상품명");
csvReader.FreeDeposit(BankType);
```

```
csvReader.PrintValue("은행", BankType, "금리계산방법", userinterest, "상품명");
csvReader.FreeSaving(BankType, userinterest);
```

```
Bank();
csvReader.YouthAccount(BankType);
```

```
CSVReader csvReader("C:/Users/chee0/Downloads/Info2.csv"); // 본인 컴퓨터에 따라 경로 바꿔줘야 함
csvReader.FinancialManual(ManualInput);
```

<csv.h>

-YouthAccount 함수

```
//사용사가 입력한 은행에 따른 청년노약계좌 정보 출력 함수
void YouthAccount( const string& value) {
    vector<vector<string>> info = GetInfo(value);
    for (const auto& cell : info[0]) {
        // 각 열의 정보 출력
     switch (&cell - &info[0][0]){
                case 0:
                    cout << '은행명: [" << cell << "]" << endl;
                    break;
                case 1:
                    coul << '기존 금리(3년 고정): [" << cell << "]" << endl;
                    break;
                case 2:
                    cout << '소득 우대금리: [" << cell << "]" << endl;
                    break;
                case 3:
                    cout << '적금담보대가산금리(반기일시상환내출): [" << cell << "]" << endl;
                    break;
                case 4:
                    cout << '직금담부대가산금리(한두대출): [" << cell << "]" << endl;
                    break;
                case 5:
                    cout << '적금담보대가산금리(대출가능한도): [" << cell << "]" << endl;
```

-FreeDeposit 함수

```
void FreeDeposit(const string& bank) {
        vector<vector<string>> info   GetInfo(bank);
        int input = 0;
            // 지장된 행들 출력
            while (true) {

                cout << "지세한 정보를 보고 싶은 상품의 번호를 입력하세요(없으면 0)" << endl;

                string inputStr;
                cin >> inputStr;

                if (inputStr == "0") {
                    break;  // 00| 입력되면 종료
                }
                if (isnumber(inputStr)    true) {
                    // stoi를 사용하여 문지열을 정수로 변환
                    input   stoi(inputStr);

                    if (input >  1 && input <  info.size()) {

                        // 입력받은 행의 열들 출력
                        for (const auto& cell : info[input - 1]) {
                            // 열에서 필요한 정보만 출력
                            if (cell != "") {
                                if ((&cell - &info[input - 1][0] <= 4) or (&cell - &info[input - 1][0] > 5))
                                    switch (&cell - &info[input - 1][0]) {
                                    case 0:
                                        coul << "은행명: [" << cell << "]" << endl;
                                        break;
```

-FreeSaving 함수

```
//사용자가 입력한 은행에 따른 적금 정보 출력 함수
void FreeSaving(const string& bank, const string &interest) {
    vector<vector<string>> info = getInfo(bank,interest,0,13);
    int input = 0;
    // 저장된 행들 출력
    while (true) {
        cout << "자세한 정보를 보고 싶은 상품의 번호를 입력하세요(없으면 0)" << endl;

        string inputStr;
        cin >> inputStr;

        if (inputStr == "0") {
            break;  // 0이 입력되면 종료
        }
        if (isnumber(inputStr) == true) {
            // stoi를 사용하여 문자열을 정수로 변환
            input = stoi(inputStr);

            if (input >= 1 && input <= info.size()) {

                // 입력받은 행의 열들 출력
                for (const auto& cell : info[input - 1]) {
                    // 열에서 필요한 정보만 출력
                    if (cell != "") {
                            switch (&cell - &info[input - 1][0]) {
                            case 0:
                                cout << "은행명: [" << cell << "]" << endl;
                                break;
```

-FinancialManual함수

```
//금융상품 정보 출력 함수
void FinancialManual(const string& name) {
    vector<vector<string>> info = GetInfo(name);
    // 저장된 행들 출력
    for (const auto& cell : info[0]) {
        if (cell != "") {
            // 각 열의 정보 출력
            switch (&cell - &info[0][0]) {
            case 1:
                cout << cell << endl;
                break;
            case 2:
                cout << "연령 조건: [" << cell << "]" << endl;
                break;
            case 3:
                cout << "소득 조건: [" << cell << "]" << endl;
                break;
            case 4:
                cout << "기간: [" << cell << "]" << endl;
                break;
```

(7) 정책/ 금융상품 선택 Main 함수

- 입출력

<financial.cpp>

yp = 클래스 YouthPolicy 변수

yf = 클래스 YouthFinancial 변수

userinput = 정책 정보와 금융상품 정보 중 출력할 정보 입력 받는 변수

- 설명

사용자가 금융 상품 정보/ 정책 정보를 선택하면 그에 맞는 함수가 호출되며

사용자가 원하는 정보를 출력하게끔 한다.

사용자가 0을 입력하면 종료 한다.

현재 많이 검색된 정책과 금융상품을 추천으로 출력한다.

- 적용된 배운 내용

클래스 : 앞서서 만든 정보 출력 클래스를 활용해 함수를 호출

조건문 : 사용자가 0을 입력하면 프로그램을 종료.

- 코드 스크린샷

<financial.cpp>

```
int main() {
    YouthPolicy yp;
    YouthFinancial yf;
    string userinput;
    while (true) {
        cout << "알고싶은 정보의 번호를 입력하세요.(종료 시 0)" << endl;
        cout << "                                    " << endl;
        cout << "1.정책 정보 2.금융상품정보" << endl;
        cout << "----------------------------" << endl;
        cin >> userinput;
        if (userinput == "0") {
            break; //0 입력 시 종료
        }
        else if (userinput == "1") {
            cout << "(수천) *중앙부처/교육   국민 취업 지원제도" << endl;
            yp.Field();
            if (yp.getFieldPolicy() == '0')   //0 입력하면 종료
                break;
            yp.Region();
            if (yp.getRegionPolicy() == "0")
                break;
            yp.PolicyShow();
        }
```

4. 테스트 결과

(1) 정부 지원 정책 데이터 수집

- 세분화

```
정책 분야를 선택하세요(종료 시 0)
--------------------------------
일자리/ 주거/ 교육/ 복지/ 참여
--------------------------------
일자리
지역을 선택하세요(종료 시 0)
-----------------------------------------------------------------------------------------
중앙부처/서울/부산/대구/인천/광주/대전/울산/경기/강원/충북/충남/전북/전남/경북/경남/제주/세종
-----------------------------------------------------------------------------------------
```

- 유효성 검증

```
1.정책 정보 2.금융상품정보
--------------------------------
s
1 또는 2를 입력해주세요
```

```
--------------------------------
일자리/ 주거/ 교육/ 복지/ 참여
--------------------------------
sl
잘못된 입력입니다. 다시 입력하세요.
```

(2) 추천 시스템

```
중앙부처
1[해외 K-Move 센터 운영]
2[국내해외취업센터 운영]
3[월드잡플러스(해외통합정보망)]
4[해외취업정착지원금]
5[민간해외취업알선]
6[고교 취업연계 장려금]
7[해외취업알선]
8[KOICA 해외사무소/재외공관 영프로페셔널(일반/사회형평적)]
9[청년일자리 도약장려금]
10[조기재취업수당]
11[선원정책 및 선원인력 역량강화 (청년해기인력 공급기반 강화)]
12[해외취업정착지원금]
13[K-Move 스쿨]
14[해외취업아카데미]
15[항공분야취업지원]
16[청년 중소기업 취업자 소득세 감면]
자세한 정보를 보고 싶은 정책의 번호를 입력하세요(없으면 0)
```

- 유효성 검증

```
4
입력한 번호의 정책이 존재하지 않습니다.
자세한 정보를 보고 싶은 정책의 번호를 입력하세요(없으면 0)
ㄴ
숫자를 입력해주세요
```

(3) 자세한 정책 정보 제공

```
1

지역: [서울]
기관 및 시사세 구분: [시사세]
정책명: [으뜸권악 청년농장]
정책 소개: [학자금 대출 주기비 비정규직 취업 등으로 어려움을 겪고 있는 청년이 밝은 미래를 준비할 수 있도록 재정적 지원]
지원 내용: [근무중인 청년에 월 10만원 또는 15만원을 2년 또는 3년간 저축하면 근무장려금 100퍼센트 매칭 지급]
시업 규모: [380명(2023 신규 120명 2022 신규 100명 2021신규 100명 2020선발 60명)]
시업운영기간: [2023. 1. ~ 12.]

사업신청기간: [2023. 5. 2.~]
연령: [만 18세 ~ 34세]
전공 요건: [제한없음]
취업 상태: [재직자]
```

(4) 은행 별 금융 상품 수집

-세분화

```
금융 상품을 선택하세요(종료 시 0)
-----------------------------------
청년도약계좌/예금/적금/금융상품설명
-----------------------------------
예금
알고싶은 예금 종류를 선택하세요(종료 시 0)
-----------------------------------------
입출금자유예금/ 정기예금
```

```
적금
알고싶은 적금 종류를 선택하세요(종료 시 0)
-----------------------------------------
정액적립식/ 자유적립식
자유적립식
금리 계산 방법을 선택하세요
-----------------------------
단리/ 복리
```

- 유효성 검증

```
금융 상품을 선택하세요(종료 시 0)
----------------------------------
청년도약계좌/예금/적금/금융상품설명
----------------------------------
s
잘못된 입력입니다. 다시 입력하세요.
```

(5) 사용자 상황 맞춤 금융 정보 제공

```
신한은행
1[신한 알.쏠 적금]
2["한 달부터 적금(매주입금)"]
```

-유효성 검증

```
자세한 정보를 보고 싶은 상품의 번호를 입력하세요(없으면 0)
3
입력한 번호의 상품이 없습니다.
자세한 정보를 보고 싶은 상품의 번호를 입력하세요(없으면 0)
s
숫자를 입력해주세요
자세한 정보를 보고 싶은 상품의 번호를 입력하세요(없으면 0)
```

(6) 구체적인 금융 상품 정보 제공

```
SC제일은행
1[e-그린세이브예금]
자세한 정보를 보고 싶은 상품의 번호를 입력하세요(없으면 0)
1
은행명: [SC제일은행]

상품명: [e-그린세이브예금]

기본 금리(%): [3.7]

최고금리(우대금리포함 %): [4]

이자지급방식: [2023-12-20]


가입방법: [인터넷뱅킹스마트뱅킹]

우대조건: ["1.SC제일은행 최초 거래 신.고객에 대하여 우대 이율을 제공함 (보너스이율0.2%)                    2.
이벤등장에서 출금하여 이 예금을 신규하는경우에 보너스이율을 제공함(기입기간:1년제/ 보너스이율:0.1% / 만기해이하
 한해 보너스이율을 적용함)"]
```

(7) 정책/ 금융상품 선택 Main 함수

```
알고싶은 정보의 번호를 입력하세요.(종료 시 0)
--------------------------------
1.정책 정보 2.금융상품정보
--------------------------------
1
(추천) *중앙부처/교육 - 국민 취업 지원제도
정책 번호를 선택하세요(종료 시 0)
```

5. 계획 대비 변경 사항

1) 정책 정보 세분화 카테고리 변경

- 이전

현재 정부에서 지원하고 있는 청년 관련 정책들 정보를 수집하여 지역별, 나이대별로 세분화함.

- 이후

현재 정부에서 지원하고 있는 청년 관련 정책들 정보를 수집하여 지역별, 분야별로 세분화함.

- 사유

본래 지역별, 나이대별을 계획했지만 청년 대상인 프로그램이기에 나이대는 중요한 정보가 아니라고 판단, 나이대별을 분야별로 대체

2) 정보 출력 방식 변경

- 이전

사용자가 더 알고자하는 정보를 클릭하면 자세한 정보와 함께 관련 사이트 연결

- 이후

사용자가 정보를 자세히 알고자 하면 세부사항 출력(사이트 정보 포함)

- 사유

본래 정보를 클릭하면 사이트가 연결될 수 있도록 하려 했으나 이는 다른 기술이 더 필요하므로 불가능하다고 판단. 사이트 정보를 출력하여 사용자로 하여금 들어갈 수 있도록 하는 형식으로 변경

3) 금융 정보 세분화 카테고리 설정

- 이전

금융 정보의 카테고리를 은행 별로 다르게 두어, 사용자가 은행을 입력하면 해당 은행의 카테고리를 나올 수 있게끔 하였음.

- 이후

청년도약계좌는 따로 둠.

- 사유

가장 대표적이고 많이 찾는 두 가지는 더 쉽게 비교할 수 있게끔 따로 빼는 게 낫다고 판단. 나머지 상품은 기타에서 은행별로 출력 예정

4) 대출 정보 대체

- 이전

사용자의 상황에 맞춘 대출 정보를 다른 금융 정보와 함께 출력

- 이후

대출 정보 대신, 이해하기 쉽도록 금융 용어 설명과 금융 상품을 세분화하여 출력함.

- 사유

적절한 대출 관련 데이터를 찾지 못하였고, 카테고리 구분이 너무 많음.

대출 특성 상 주택담보대출, 전세자금대출, 월세보증금 대출, 개인 신용대출 등 종류가 굉장히 많고, 은행 별로 각 대출상품마다 조건이 상이함. 이 때문에 은행 사이트에 있는 데이터를 개별로 가져와야 하는데, 오픈되어 있는 대출 상품 데이터를 찾지 못함. 또한 분류가 복잡하기 때문에 코드도 더욱 복잡해질 것을 우려하여 대출 정보를 제거함.

대신, 대학생들이 접근하기 쉽도록 금융 용어 설명을 넣고, 예/적금 으로만 나누어져 있던 금융 상품 카테고리를 다양화함.

5) 추천 알고리즘 대체

- 이전

사용자의 입력이 많은 순으로 상단에 추천 상품 출력

- 이후

현재(2023.12.24) 기준으로 가장 많이 찾는 정책과 금융 상품을 추천으로 출력.

- 사유

다른 사이트의 사용자 검색 데이터를 가져오는 것은 불가능, 현재 프로그램 안에서의 입력으로 처리하기에는 프로그램을 다시 시작할 때마다 리셋되어 의미가 많지 않다고 판단.

대신 현재 기준, 인터넷에서 가장 많이 찾는 정책과 금융 상품을 추천으로 띄움.

(온통청년 사이트 참고)

6. 느낀점

간단한 프로그램이라고 생각했는데, 생각보다 고려해야 할 게 많아서 시간이 오래 걸렸습니다. 배운 것들을 활용해서 최대한 하려고 해보았는데, 코드가 길어질 수록 계속해서 얽혀서 오류가 났을 때 원인이 무엇인지 정확하게 파악하는 과정이 가장 어려웠던 것 같습니다. 수업시간에 배운 것들을 최대한 활용하려고 하였지만, 후반에 배운 것들(상속 등)을 능숙하게 해내지 못해서 그걸 적용하지 못한 게 아쉽습니다.

과제가 조금 많게 느껴질 때도 있기는 했지만, C++ 을 처음 배우는 수업이었는데 너무 어렵지 않게 처음부터 잘 알려주셔서 얻어가는게 많은 수업이었습니다. 언어 외에도 깃허브 사용법이나 코드 스타일 등 여러가지를 알려주셔서 도움이 많이 되었습니다.

한 학기동안 정말 수고 많으셨습니다!