HW5 실습

Make 환경 실습 & Polynomial

Make 환경

- Make
  - 코드 컴파일, 실행 파일 빌드 및 문서 생성 등과 같은 복잡한 프로세스를 자동화하는 데 도움을 주는 빌드 자동화 도구

- Makefile
  - 소스 코드 파일과 해당 종속성, 빌드 명령 및 변수와 같은 빌드 프로세스의 규칙을 정의한 텍스트 파일
  - make는 makefile에 적힌 규칙과 종속성을 기반으로 빌드 프로세스를 순차적으로 진행

Make 환경의 장점

- 시간 절약
  - 빌드에 필요한 반복 작업을 최소화, 빌드 프로세스 자동화 등을 통한 빠른 빌드 가능
- 의존성 관리
  - 파일 간 종속구조 빠르게 파악 가능
- 플랫폼 종속성
  - Unix와 Unix-like 시스템의 표준 빌드 도구로 사용되어 소프트웨어 프로젝트를 다양한 운영 체제와 환경에서 구축, 재사용 가능

설치

- Mac
  - 설치 불필요
- Windows
  - 실습1을 잘 따라왔다면 g++ 설치 시에 자동으로 설치됨

기본 구조

```
# 매크로 정의부
CC = g++ # 컴파일러 지정

target_a: dependency1 dependency2
    command1
    command2

target_b: dependency1 dependency2 dependency3
    command1
    command2
    command3
```

- Target
  - 생성하려는 파일/작업
  - 빌드하고자 하는 대상
- Dependency(prerequisites)
  - Target을 생성하기 위해 필요한 파일/작업
  - 필요한 모든 종속 항목을 정의
- Command(Recipes)
  - target을 생성하기 위한 명령어
  - **명령어는 공백문자가 아닌 tab 문자로 시작**

실습

- addition.h

```h
#ifndef ADDITION_H
#define ADDITION_H
int add(int a, int b);
#endif
```

- addition.cpp

```cpp
#include "addition.h"
int add(int a, int b)
{ return a + b; }
```

- main.cpp

```cpp
#include <iostream>

#include "addition.h"
int main() {
  int num1 = 5;
  int num2 = 10;
  int result = add(num1, num2);
  std::cout << "add result: " << result
   << std::endl;
  return 0;
}
```

makefile(1)

```
# 변수 선언
CC=g++
CFLAGS= -o # 컴파일러 옵션: '-o'
OBJS=main.o addition.o # 오브젝트 파일 목록
TARGET=main # 생성할 실행 파일

# 프로그램 실행 타겟팅
run: all
	./$(TARGET)
# 실행 파일 만드는 타겟팅
all: $(TARGET)
```

makefile(2)

```
# make clean(빌드된 오브젝트 파일과 실행파일 삭제) 명령어를 수행
clean:
	rm -rf *.o
	rm -rf $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) $@ $(OBJS)

# 의존성 오브젝트 생성: makefile 수정여부 추적하고 다시 컴파일할 때 필요
main.o: addition.h main.cpp
addition.o: addition.h addition.cpp
```

결과

```
~/school/Master/M2/DS/ex5/ make
c++    -c -o main.o main.cpp
c++    -c -o addition.o addition.cpp
g++ -o  main main.o addition.o
./main
add result: 15
~/school/Master/M2/DS/ex5/ make clean
rm -rf *.o
rm -rf main
~/school/Master/M2/DS/ex5/ make all
c++    -c -o main.o main.cpp
c++    -c -o addition.o addition.cpp
g++ -o  main main.o addition.o
~/school/Master/M2/DS/ex5/ make run
./main
add result: 15
```

터미널 단축키 : ctrl + ~

```
D:\SCHOOL\2025-2\ex5>mingw32-make
g++ -o  main main.o addition.o
./main
add result: 15
```

만능 makefile(참고)

```
CC = g++

# C++ 컴파일러 옵션
CXXFLAGS = -Wall -O2

# 링커 옵션
LDFLAGS =

# 소스 파일 디렉토리
SRC_DIR = ./src

# 오브젝트 파일 디렉토리
OBJ_DIR = ./obj

# 생성하고자 하는 실행 파일 이름
TARGET = main

# Make 할 소스 파일들
# wildcard 로 SRC_DIR 에서 *.cc 로 된 파일들 목록을 뽑아낸 뒤에
# notdir 로 파일 이름만 뽑아낸다.
# (e.g SRCS 는 foo.cc bar.cc main.cc 가 된다.)
SRCS = $(notdir $(wildcard $(SRC_DIR)/*.cc))

OBJS = $(SRCS:.cc=.o)

# OBJS 안의 object 파일들 이름 앞에 $(OBJ_DIR)/ 을 붙인다.
OBJECTS = $(patsubst %.o,$(OBJ_DIR)/%.o,$(OBJS))
DEPS = $(OBJECTS:.o=.d)

all: main

$(OBJ_DIR)/%.o : $(SRC_DIR)/%.cc
    $(CC) $(CXXFLAGS) -c $< -o $@ -MD $(LDFLAGS)

$(TARGET) : $(OBJECTS)
    $(CC) $(CXXFLAGS) $(OBJECTS) -o $(TARGET) $(LDFLAGS)

.PHONY: clean all
clean:
    rm -f $(OBJECTS) $(DEPS) $(TARGET)

-include $(DEPS)
```

- 프로젝트 구조는 다음과 같아야함

```
$ tree
.
├── Makefile
├── obj
└── src
    ├── bar.cc
    ├── bar.h
    ├── foo.cc
    ├── foo.h
    └── main.cc
```

Sort algorithm

- C++의 algorithm 헤더에 포함
- 기본적으로 오름차순 정렬 수행
- Compare함수를 만들어 해당 함수의 반환 값에 맞게 정렬이 동작하도록 설정 가능

```
#include <algorithm>
#include <iostream>
using namespace std;
void Print(int *arr) {
  cout << "arr[i] : ";
  for (int i = 0; i < 10; i++) {
    cout << arr[i] << " ";
  }
  cout << endl;
}
bool compare(int a, int b) { return a > b; }
int main(void) {
  int arr[10] = {3, 7, 2, 4, 1, 0, 9, 8, 5, 6};
  int arr2[10] = {3, 7, 2, 4, 1, 0, 9, 8, 5, 6};
  Print(arr);  // 정렬 전 출력

  sort(arr, arr + 10);  // default(오름차순)로 정렬
  Print(arr);           // 정렬 후 출력

  sort(arr2, arr2 + 10, compare);  // compare(내림차순)로 정렬
  Print(arr2);                     // 정렬 후 출력
  return 0;
}
```

실습 설명

- 기술서에 적혀있는 대로 polya.cpp 나 polyb.cpp의 비어있는 부분을 작성할 것

실습 설명

```
ostream& operator<<(ostream& os, Polynomial& p) {
  // .....
  return os;
}
void Polynomial::NewTerm(const float theCoeff, const int theExp) {
  // .....
}
Polynomial Polynomial::operator+(Polynomial& b) {
  // .....
}
Polynomial Polynomial::operator*(Polynomial& b) {
  Polynomial c;
  //.....
  return c;
}
```

```
[B611107@localhost hw2]$ hw2a < poly.in
x^8 -7x^5 -x^3 -3
x^5 +2x^3 -4
x^8 -6x^5 +x^3 -7
```

질문

- pemds81718@gmail.com
- 간단한 구글링으로 알 수 있는 내용은 답변하지 않습니다.