Code Slayer

프로젝트 중간보고서

컴퓨터공학과 | **윤건호** 팀장

컴퓨터공학과 | **안치원** 팀원

TEAM 7

팀원 **김상원** | 컴퓨터공학과

팀원 **우은혁** | 컴퓨터공학과

- **과목명** : C++프로그래밍
- **담당교수** : 안용학
- **수업시간** : 13:30 ~ 15:00
- **제출일자** : 2020. 05. 28.

1. 문서 개요

가. 프로젝트 개요

- **프로그램명 : CodeSlayer**
- **개발 배경 :**
  (1) 영어 코드 타자가 익숙하지 않은 사용자의 타자 실력 증진
  (2) C++ 언어의 키워드를 활용하여 사용자의 코딩 생산성 증진
  (3) 위 요소와 미니 게임을 결합하여 흥미 유발
- **주요 기능 :**
  (1) C++ 코드 타자 연습
  (2) C++ 코드 빈칸 추론 게임

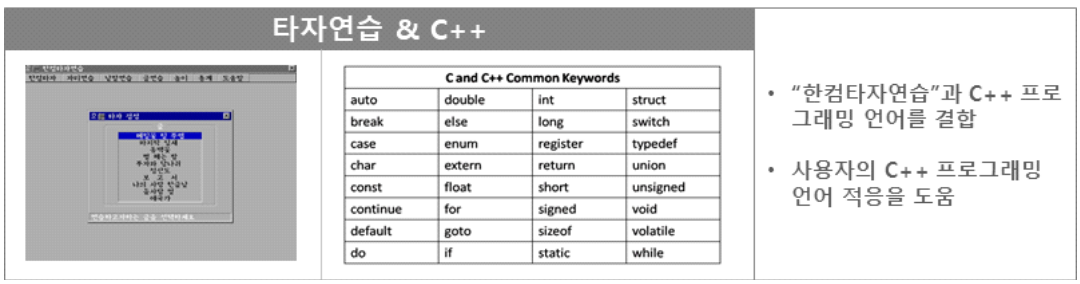
*** CodeSlayer ***

나. 목차

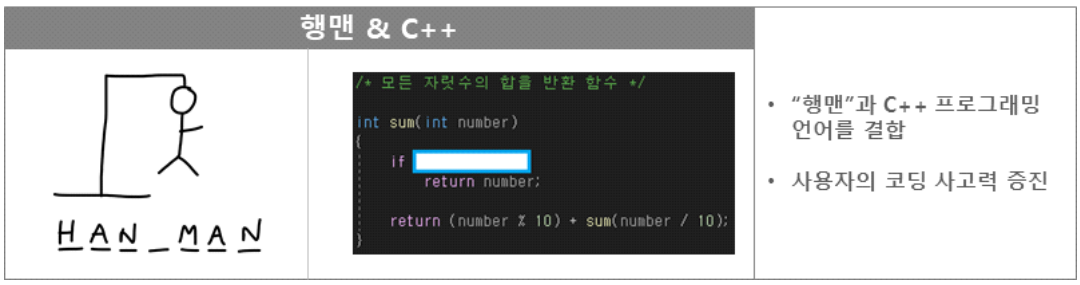
2. 개발 목표

가. 최종 개발 목표

사용자의 코딩 생산성 향상을 최종 개발 목표로 한다. 이를 달성하기 위해, 크게 두 가지의 개발 목표를 설정하였다. **첫째, "한컴타자연습"에서 착안한 타자 연습 기능이다.** "한컴타자연습"에서는 단어연습, 짧은 글 연습, 긴 글 연습을 활용하여 사용자의 한글 타자 실력 향상을 돕는다. 이 프로그램에서는 "한글" 대신 "영어" 또는 "C++ 키워드 또는 코드"를 활용한다. 이를 통해 사용자는 영어, C++ 키워드, 그리고 코드에 익숙해질 수 있다. 이는 곧 코딩 생산성의 향상으로 이어진다. **둘째, "행맨"에서 착안한 미니 게임 기능이다.** "행맨"은 플레이어가 숨겨진 철자를 추측해가며, 최종 단어를 맞히는 게임이다. 이 프로그램에서는 "행맨"의 기존 숨겨진 철자 대신, 특정 코드의 가려진 부분을 맞히는 형식으로 제작한다. 이를 통해 사용자는 코딩에 흥미를 느낄 수 있으며, 추측하는 과정을 통해 코딩 사고력을 증진할 수 있다.



*** 타자 연습과 C++ 프로그래밍 언어를 결합한 코드 타자 연습 ***



*** 행맨과 C++ 프로그래밍 언어를 결합한 미니 게임 ***

나. 세부 개발 목표

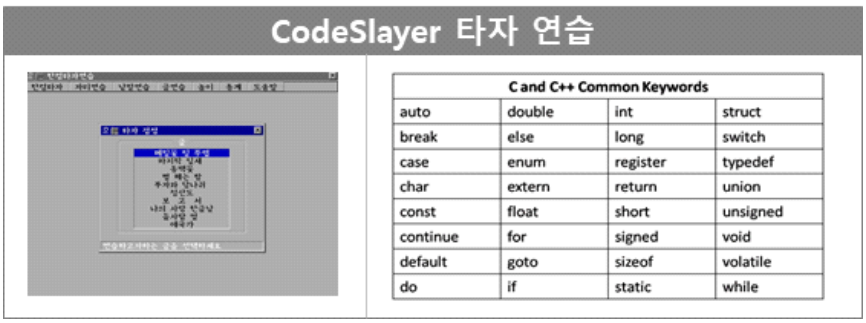
1) 타자 연습 : 코드 타자 연습 기능 제공

사용자가 **C++ 코드를 빠르고 정확하게 작성**하도록 돕는다. 기본적인 틀은 다음과 같다. 첫째, 사용자에게 입력할 텍스트(C++ 키워드, 코드, 문장 등)를 출력한다. 둘째, 사용자는 주어진 텍스트를 최대한 빠르고 정확하게 입력한다. 셋째, 프로그램은 주어진 텍스트를 사용자가 얼마나 빠르고 정확하게 입력하였는지 측정한다. 마지막으로, 측정한 결과를 활용하여 사용자의 정확도와 타자 속도를 기록한다. 기록은 추후 열람할 수 있으며, 사용자는 자신이 얼마나 발전하였는지 확인할 수 있다. 텍스트의 길이 또는 사용자의 목적에 따른 다음 세 가지 기능을 제공한다.

(1) 단어연습 : 텍스트 파일에서 무작위로 추출한 단어 하나가 주어진다. 단어에는 C++ 키워드, 프로그램에서 사용한 변수명 등이 포함된다.

(2) 짧은 글 연습 : 텍스트 파일에서 무작위로 추출한 문장 하나가 주어진다. 문장에는 간단한 코드, C++에서 자주 사용하는 함수명 등이 포함된다.

(3) 긴 글 연습 : 무작위 텍스트 파일에서 추출한 단락이 주어진다. 단락에는 일반적인 C++ 코드, 프로그램에서 사용한 함수 등이 포함된다.

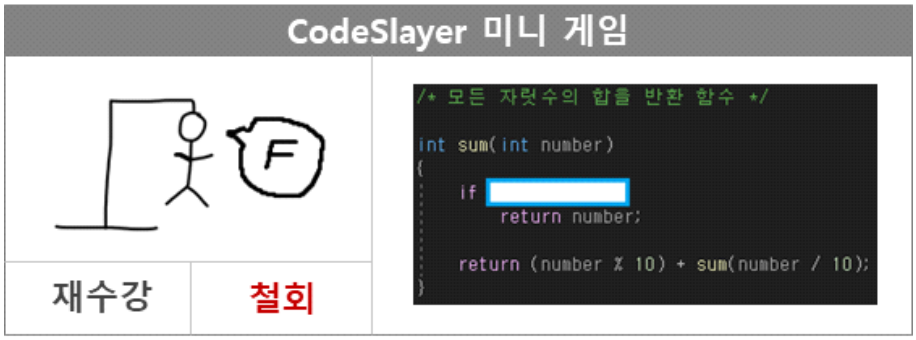


*** C++ 타자 연습 ***

2) 미니 게임 : "행맨"과 코딩을 결합한 미니 게임

CodeSlayer는 타자 연습뿐만 아니라, C++과 "행맨"을 결합한 새로운 게임을 제공한다. 이를 활용하여 사용자의 **코딩 사고력 증진**을 돕는다. 기본적인 틀은 다음과 같다. 첫째, 빈칸이 뚫려 있는 코드와 코드의 목적을 출력한다. 둘째, 사용자는 코드의 목적을 바탕으로 빈칸을 채워 넣는다. 셋째, 프로그램은 사용자가 입력한 코드의 답을 판별한다. 오답이 많을수록, 교수대 그림이 완성돼가고 사용자의 학점(목숨)은 떨어진다. 교수대 그림이 완성됨과 동시에 학점이 F가 되면, C++ 과목을 철회하면서 패배한다. "행맨"의 상세 규칙은 다음과 같다.

"행맨" : 행맨은 영어 단어 맞히기 게임 중 하나이다. 처음에는 글자 수만큼의 빈칸과 사람이 없는 교수대가 출력된다. 플레이어는 26개(영어 기준)의 철자 중 하나를 선택한다. 그 철자가 단어에 포함되어 있다면, 해당하는 빈칸에 철자가 채워진다. 만약 제시한 철자가 단어에 포함되어 있지 않다면, 교수대 아래에 밧줄, 머리, 팔, 손, 몸통, 다리, 발 순서로 사람이 그려진다. 그림이 완성되면 게임에서 패배하고, 단어를 완성하거나 알아내면(규칙에 따라 다름) 게임에서 승리한다.

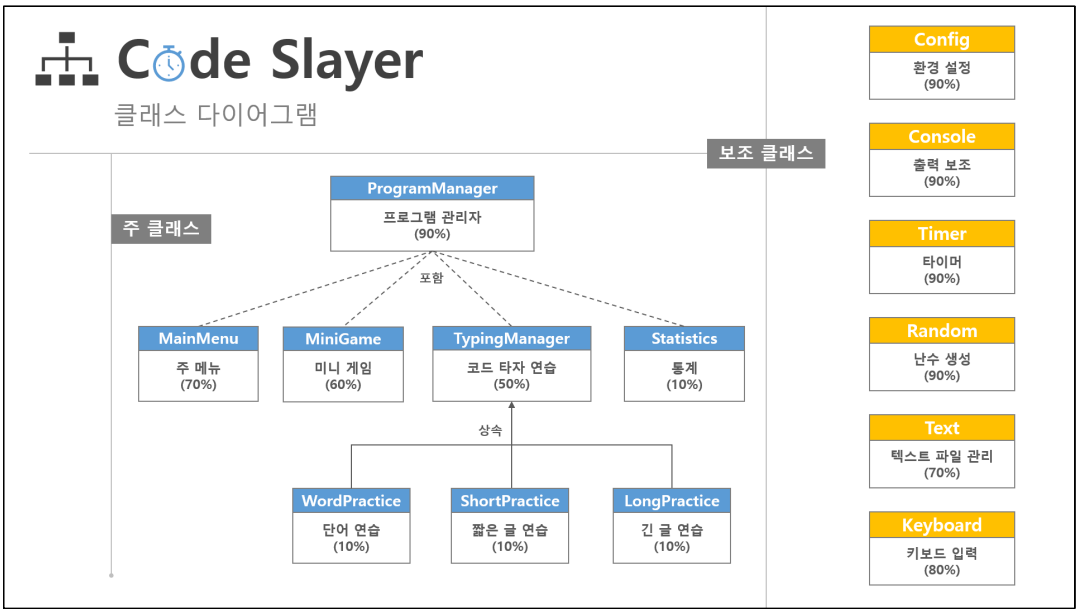


*** C++ 미니 게임 ***

3. 진행 상황

가. 개요

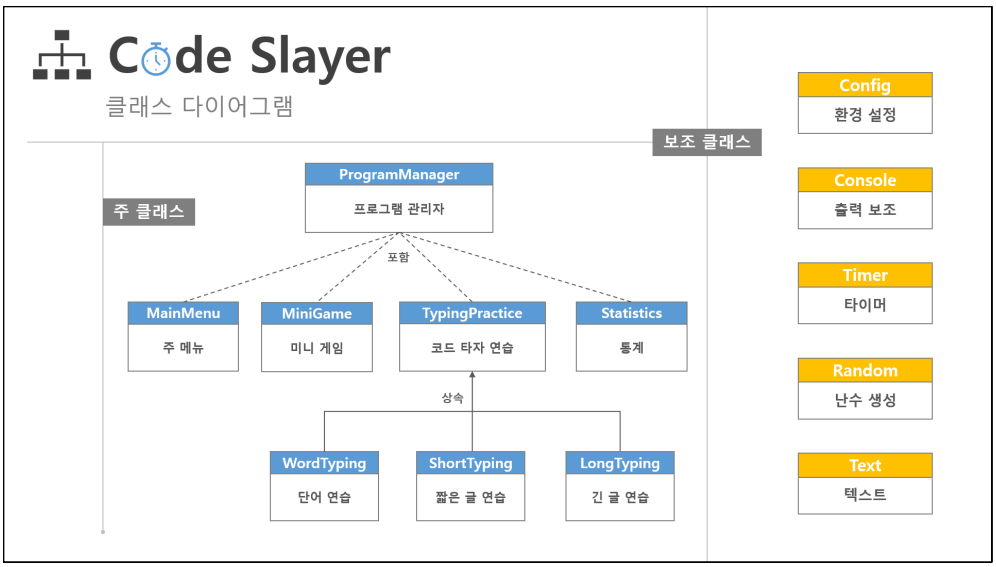
프로젝트의 **진행 상황을 서술**한다. 가장 먼저 현재의 진행 상황을 나타낸 클래스 다이어그램을 살펴본다. 그리고 제안서에 있던 기존 클래스 다이어그램과 현재의 클래스 다이어그램을 비교한다. 마지막으로, 단기 개발 목표와 장기 개발 목표를 서술하여 앞으로의 개발 방향을 제시한다.



*** 프로그램 구조와 진행 상황 요약 ***

나. 클래스 다이어그램 (제안서)

클래스 다이어그램을 활용하여 프로그램의 **전체 클래스 구조**를 설명한다. 이 항목에 있는 클래스 다이어그램은 이전 제안서에 첨부되었던 버전이며, 중간보고서와 비교하기 위해 삽입하였다. 프로그램의 전체적인 흐름은 최상단에 있는 ProgramManager 클래스가 관리한다. 그리고 MainMenu, TypingManager, MiniGame, Statistics 클래스는 각각 주메뉴, 코드 타자 연습, 미니 게임, 통계 기능을 담당한다. ProgramManager에서는 네 가지 기능 중 사용자가 원하는 기능을 호출하여 해당 객체의 루프를 실행한다. TypingPratice는 세부적으로 단어연습, 짧은 글 연습, 긴 글 연습의 세 가지 유형으로 나누어진다.

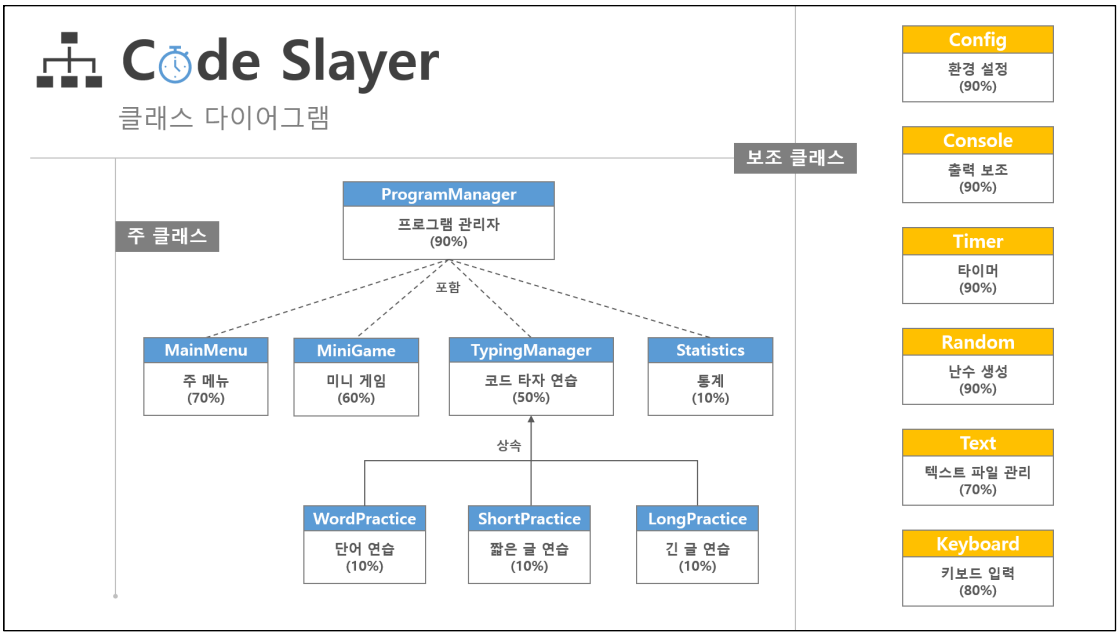


*** (제안서) 클래스 다이어그램 ***

다. 클래스 다이어그램 (최신)

현재 프로그램의 구조를 클래스 다이어그램으로 나타내었다. 이전의 클래스 다이어그램과 비교하였을 때 키보드 입력을 담당하는 보조 클래스 Keyboard가 추가되었다.

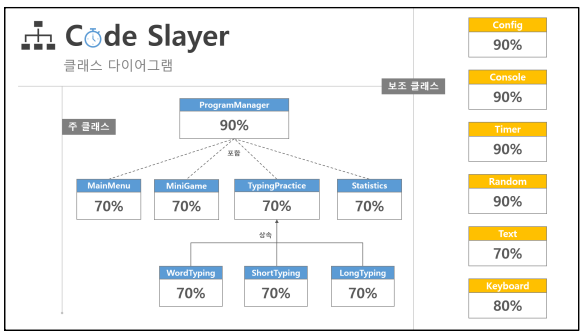
각 클래스에는 개발 진행 상황을 나타내는 퍼센티지를 표기하였다. 진행 상황 퍼센티지는 일반적으로 다음과 같다. 90% 이상은 기본 틀 제작 완료 및 활용 중 80~89%는 기본 틀 제작 완료 및 내용 추가 중, 70~79%는 기본 틀 기능 추가 중, 70% 미만은 기본 틀 제작 중임을 나타낸다.



*** (최신) 클래스 다이어그램 ***

라. 단기 목표

모든 클래스의 **기본 틀을 완성**하는 것이 최우선 목표이다. 현재 대부분의 보조 클래스는 응용 단계까지 구현하였으나, 주 클래스 중 코드 타자 연습과 그의 파생 클래스, 통계 클래스, 미니 게임 클래스는 기본 틀이 완성되지 않았다. 따라서 주 클래스의 기본 틀 제작을 최우선 목표로 설정하였다. 기본 틀 제작이 완료되어 테스트가 가능해지면 다음 단계로 넘어간다.

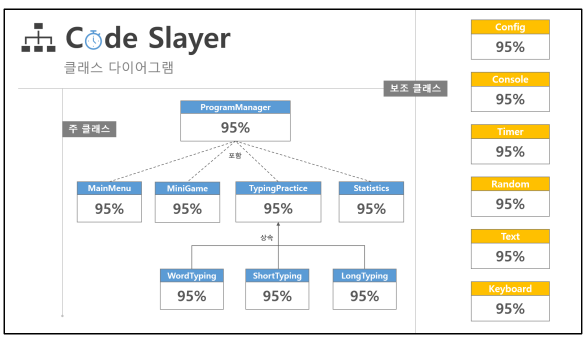


*** 단기 목표 ***

마. 장기 목표

모든 클래스의 기본 틀을 개발하면 각 **클래스와 프로그램의 완성**도를 높인다. 먼저 기능 추가, 레이아웃 수정, 프리셋 데이터 추가, 시나리오 테스트, 코드 성능 개선 등의 방법을 사용하여 클래스의 완성도를 높인다.

최종적으로는 프로그램의 완성도를 높여 사용자에게 더 나은 경험을 제공한다.



*** 장기 목표 ***

4. 개발 내용

가. 개요

설계한 클래스들이 프로그램에서 어떤 기능을 수행하는지, 즉 **클래스의 역할**을 서술한다. 보조 클래스는 어느 클래스에서나 포함하여 사용할 수 있으므로, 가장 먼저 보조 클래스의 기능을 살펴본다. 그리고 주 클래스의 기능과 역할을 확인한다. 마지막으로 개발한 내용을 테스트한다.

각 클래스의 명세서에는 클래스 개요, 개발 완료 내용, 개발 예정 내용이 포함된다. 개발이 거의 완료되었거나 테스트 중인 클래스는 지금까지 개발한 내용을 중심으로 서술한다. 아직 개발되지 않았거나 개발이 미흡한 클래스는 앞으로의 개발 방향을 중심으로 서술한다.

* CodeSlayer *

나. 보조 클래스

Config	Console	Timer	Random	Text	Keyboard
환경 설정 (90%)	출력 보조 (90%)	타이머 (90%)	난수 생성 (90%)	텍스트 파일 관리 (70%)	키보드 입력 (80%)

* 보조 클래스 *

Config
환경 설정
(90%)

```
/* 파일 개수 */
class FileNum
{
public:

    static const unsigned int Word = 50;
    static const unsigned int Short = 25;
    static const unsigned int Long = 10;
    static const unsigned int Game = 17;
};

/* 스크린 */
enum class Screen
{
    MainMenu,
    WordPractice,
    ShortPractice,
    LongPractice,
    MiniGame,
    Exit,
};
```

1) Config | 환경 설정

개요 : 프로그램에서 사용하는 **상수를 저장하고 관리**한다. 상수에는 프리셋 파일의 개수, 스크린의 번호 등이 포함된다. 프로그래머가 쉽게 값을 수정할 수 있고, 모든 클래스에서 접근할 수 있으므로 프로그램 관리가 편리해진다.

완료 : 첫째, 프리셋 텍스트 파일의 개수를 저장한다. 이 값을 참고함으로써, 텍스트 파일을 불러올 때 오류를 방지할 수 있다. 둘째, 열거형 스크린 클래스를 정의한다. 정수가 아닌 문자열의 형태이므로, 프로그래머가 직관적으로 이해할 수 있다. 이러한 환경 상수들을 프로그램의 주 클래스에서 사용하고 있다.

예정 : 필요에 따라 기존의 상수를 수정하고 새로운 상수들을 추가한다. 여러 개의 상수 클래스를 하나의 클래스로 묶어 완성도를 높이는 방법 또한 고려하고 있다.

Console
출력 보조
(90%)

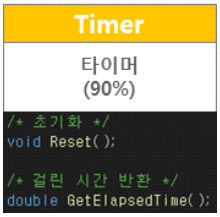
```
/* 커서 위치 지정 */
void CursorPosition(int x, int y);

/* 텍스트 색 지정 */
void Color(const string& foregroun

/* 콘솔 창 비우기 */
void Clear();

/* 콘솔 창 일부 비우기 */
void Clear(int startX, int startY,

/* 문자열 또는 파일 출력 */
void Draw(string str_or_file, cons
```

2) Console | 출력 보조

개요 : **콘솔 창의 출력을 보조**한다. 커서의 좌표 지정, 글자색 지정, 화면 비우기, 텍스트 파일 출력하기 등의 기능을 추상적으로 제공하여 전체 코드의 길이를 줄일 수 있다.

완료 : 크게 네 가지 기능을 제공한다. 첫째, 콘솔 창 커서의 좌표를 지정할 수 있다. 다음 콘솔 창 출력은 해당 좌표에서 이루어진다. 둘째, 출력할 문자의 글자색과 배경 색을 지정할 수 있다. 다른 색을 지정해주기 전까지는 해당 색으로 출력한다. 셋째, 콘솔 창을 비운다. 전체 콘솔 화면을 비우거나 사용자가 지정한 크기만큼 비울 수 있다. 넷째, 종합하여 출력한다. 위 세 가지 기능과 더불어 텍스트 파일 출력 또한 지원한다. 네 가지 기능을 활용하여 코드의 길이를 크게 줄일 수 있었다.

예정 : 각 클래스에서 활용하여 프로그램의 코드 길이를 줄인다.



3) Timer | 시간 측정

개요 : 걸린 **시간을 측정**한다. 일반적인 스톱워치와 마찬가지로, 타이머 초기화 및 걸린 시간 반환 기능을 제공한다. 각 타이머 객체는 서로에게 영향을 미치지 않는다.

완료 : 두 가지 기능을 제공한다. **첫째,** 타이머를 초기화한다. 초기화한 시점의 시각을 0으로 설정한다. **둘째,** 걸린 시간을 측정한다. 초기화한 시점으로부터 측정 함수를 호출할 때까지 걸린 시간을 초 단위로 반환한다.

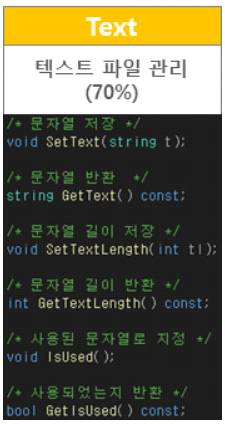
예정 : 각 클래스에서 필요할 때 사용한다.

4) Random | 무작위 난수 생성

개요 : 범위 내 **무작위 난수를 반환**한다. 고품질의 난수를 쉽게 얻을 수 있다.

완료 : 사용자가 지정한 닫힌 구간 [a, b] 내의 무작위 정수를 반환한다. C++ 표준에서 제공하는 하드웨어 시드와 메르센-트위스터 19937 알고리즘을 활용하여 높은 품질의 난수를 얻을 수 있다. 무작위 텍스트 파일을 선택할 때 응용하고 있다.

예정 : 필요하다면 무작위 실수를 반환하는 함수를 추가한다.



5) Text | 텍스트 관리

개요 : 프로그램에서 사용하는 **프리셋 텍스트를 관리**한다.

완료 : Text 객체는 크게 두 가지의 기능을 갖는다. 첫째, 문자열을 저장한다. 파일에서 읽어온 문장 또는 단어로 이루어진 문자열을 객체 단위로 저장한다. 둘째, 해당 문자열의 사용 여부를 확인한다. 사용된 문자열과 사용되지 않은 문자열을 구분함으로써, 같은 문제를 중복으로 출력하지 않을 수 있다.

예정 : 주 클래스의 개발과 함께 지속해서 보완한다. 특히 타자 연습과 미니 게임을 구현하는 과정에서 추가해야 할 기능이 많을 것으로 예상된다.

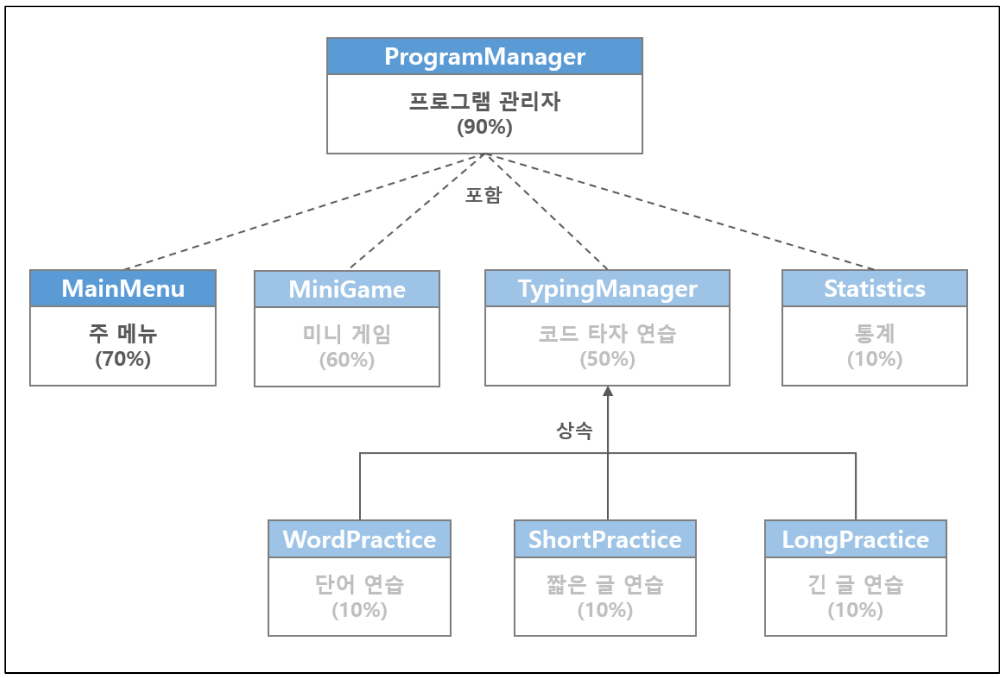
6) Keyboard | 키보드 입력 보조 (추가)

개요 : **키보드의 입력을 처리**한다. 키보드 입력과 키 상수들을 각 클래스에서 담당하는 것이 아니라, Keyboard 클래스에서 담당하므로 프로그램의 유지보수가 쉬워진다.

완료 : 모든 키보드 입력을 문자열과 정수의 쌍으로 일대일 결합한다. 따라서 프로그래머는 원하는 키의 이름을 문자열 인자로 전달해주면, 클래스 내에서 정수로 치환하여 처리한다. 크게 세 가지의 기능을 구현하였다. **첫째,** 정적 입력이다. 사용자의 키보드 입력이 있을 때까지 대기한다. **둘째,** 동적 입력이다. 사용자의 입력이 없다면 대기하지 않고 프로그램을 계속 진행한다. **셋째,** 특정 키가 눌렸는지 확인한다. 이때 프로그래머는 원하는 키를 문자열 인자로 전달하여 특정 키가 눌렸는지 아닌지를 반환받을 수 있다. 화살표 키를 이용하여 메뉴를 이동할 때 응용하고 있다.

예정 : 사용이 끝나면 마지막으로 입력된 문자를 초기화해야 한다. 따라서 필요한 구간에서의 사용이 끝난 후 프로그래머가 Clear 함수를 명시적으로 호출해야 한다. 이를 자동화하는 방법을 구상하고 있다.

다. 주 클래스



*** ProgramManager & MainMenu ***

ProgramManager
프로그램 관리자
(90%)

```
private:

    Console* mConsole;
    Keyboard* mKeyboard;
    Random* mRandom;
    TypingManager* TypingManager;
    MainMenu* mMainMenu;

    WordPractice* mWordPractice;
    ShortPractice* mShortPractice;
    LongPractice* mLongPractice;
    MiniGame* mMiniGame;

    Screen mCurrentScreen;

    bool mQuit;

    /* 소개 화면 */
    void IntroScreen();

    /* 종료 화면 */
    void ExitScreen();


public:

    ProgramManager();
    ~ProgramManager();

    /* 프로그램 메인 루프 */
    void MainLoop();
```

1) ProgramManager | 프로그램 관리

개요 : 전체적인 **프로그램의 흐름을 관리**한다.

완료 : 필요한 객체들을 프로그램이 시작될 때 생성한다. 메인 루프의 순서는 다음과 같다. 가장 먼저 소개 화면을 제공한다. 팀과 팀원, 그리고 프로그램 로고를 출력한다. 사용자가 엔터를 입력하면 메인 메뉴로 진입한다. 메인 메뉴에서는 화살표와 엔터 키를 이용하여 각 기능으로 이동할 수 있다. 따라서 사용자가 원하는 기능(스크린)을 호출한다. 메인 메뉴, 미니 게임, 타자 연습, 종료 등의 기능이 포함된다. 마지막으로 사용자가 프로그램을 종료하면 종료 화면을 출력하고, 생성한 모든 객체를 소멸시키고, 프로그램을 종료한다.

예정 : 다른 클래스의 진행 상황에 따라 내용을 추가한다. 그리고 종료 화면의 레이아웃을 다듬는다.

* 좌측 코드는 헤더 파일의 일부
* 실제 개발 진행 상황은 상단 퍼센티지 참고

MainMenu
주 메뉴
(70%)

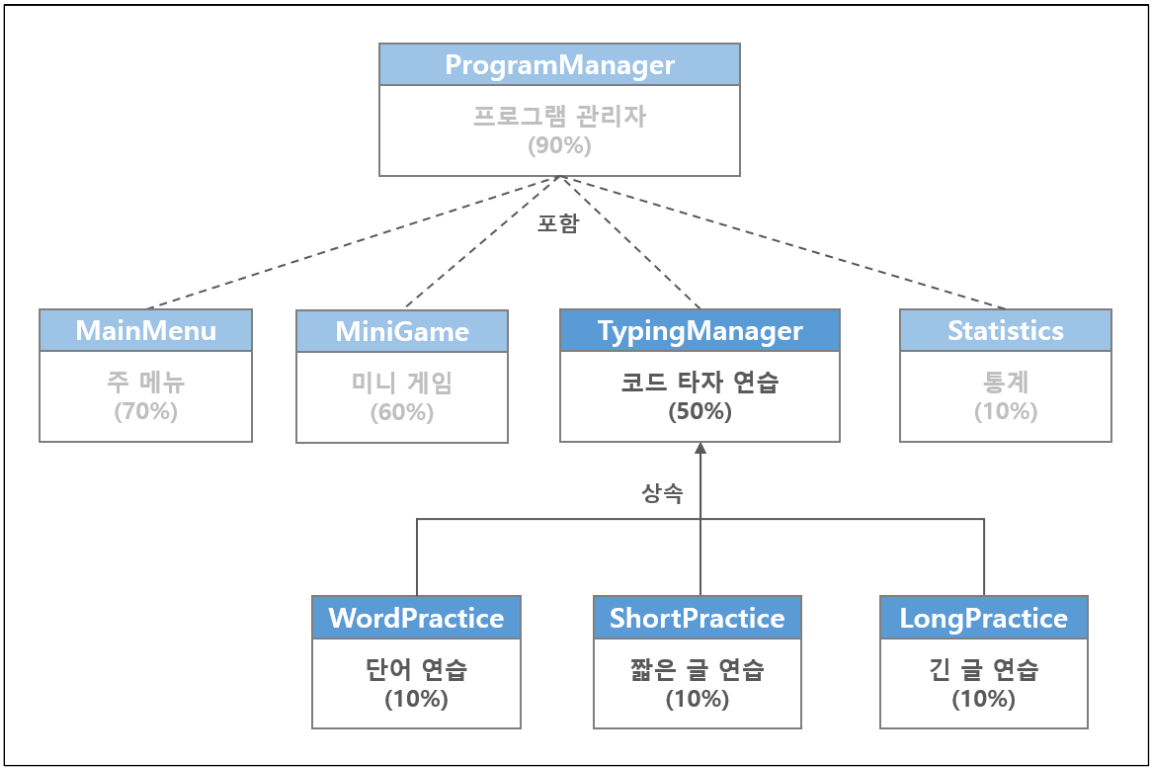
```
    /* 화면 출력 */
    void Render();

public:

    MainMenu();
    ~MainMenu();

    /* 메인 메뉴 루프 */
    void Main();
```

2) MainMenu | 메인 메뉴

개요 : 프로그램의 **메인 메뉴를 제공**한다.

완료 : 사용자에게 메인 메뉴를 출력하고 원하는 기능을 입력받는다. 가장 먼저 메인 메뉴 레이아웃을 출력한다. 사용자는 키보드를 활용하여 원하는 기능을 선택한 후 엔터를 누른다. 엔터가 입력되면 메인 메뉴는 종료되고 ProgramManager로 다음 스크린의 열거형 값을 반환한다. ProgramManager는 MainMenu 객체에서 반환된 스크린 값을 바탕으로 다음 기능을 실행한다. 스크린의 값은 Config에 선언되어 있다.

예정 : 다른 주 클래스들의 기본 틀이 완성되면, 메인 메뉴에서 선택할 수 있도록 선택지를 추가한다. 레이아웃을 다듬어 완성도를 높인다.



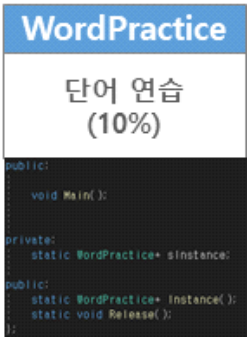
*** TypingManager 등 4개 ***

3) TypingManager | 타자 연습 공통

개요 : 모든 **타자 연습 클래스에서 사용하는 공통 기능을 제공**한다.

완료 : 가장 기본적인 변수와 기능들을 구현하였다. 먼저 존재하는 모든 프리셋 텍스트 파일을 불러온다. 불러온 텍스트 파일을 Text 객체에 저장하여 벡터에 저장한다. 멤버 변수 중 각 텍스트 벡터는 포인터 벡터로 만들어 오직 하나의 벡터를 가리킨다. 따라서 메모리 누수를 최소화하였다. 그리고 각 파생 클래스(단어, 짧은 글, 긴 글)에서 사용하는 공통 기능들을 구현하였다. 무작위 문자열 추출, 평균 타자 속도 계산, 소요 시간 측정, 정답 판별 등의 기능이 포함된다.

예정 : 최우선으로 개발한다. 이 클래스의 기본 틀이 완성되어야, 상속받는 클래스들 또한 개발할 수 있기 때문이다. 그리고 파생 클래스와 함께 지속해서 업데이트한다

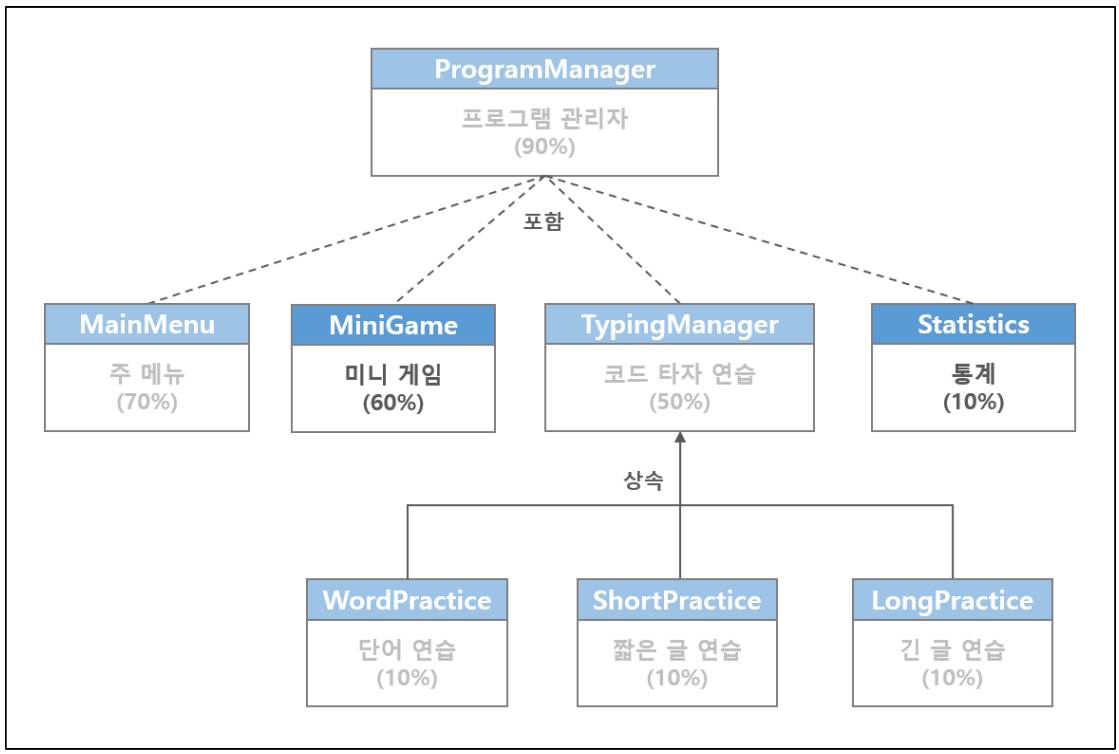


4) WordTyping 등 3개 | 코드 타자 연습

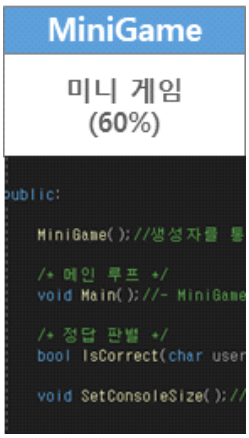
개요 : 사용자가 원하는 유형의 **타자 연습 기능을 제공**한다.

완료 : 상위 클래스인 TypingManager의 기본 틀이 완성되면 개발할 예정이다. 따라서 아주 기본적인 틀만 제작한 상태이다.

예정 : ProgramManager에서 해당 객체의 Main을 호출하여 사용한다. 흐름은 다음과 같다. 첫째, 한 문항을 불러온다. 둘째, 사용자의 입력을 받아 타자 속도 등의 변수를 업데이트한다. 셋째, 무작위의 다음 문항을 불러온다. 이 과정을 일정 횟수 반복한 후, 기록을 출력하고 파일에 저장한다. 부모 클래스인 TypingManager의 기본 틀이 완성되면 개발할 예정이다. TypingManager과 함께 지속해서 보완하여 완성도를 높인다.



* MiniGame, Statistics *

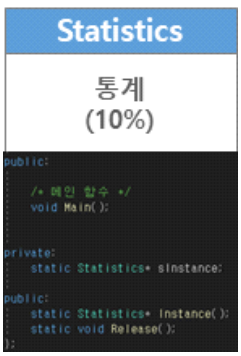


5) MiniGame | 코드 빈칸 추론 게임

개요 : "C++을 "행맨"을 결합한 **코드 빈칸 추론 게임을 제공**한다.

완료 : ProgramManager에서 호출하여 실행할 수 있다. 기본적인 흐름은 다음과 같다. 첫째, 행맨의 레이아웃과 목숨(학점)을 출력한다. 둘째, 문제를 출력하고 사용자의 입력을 받는다. 셋째, 학점이 F가 되거나 모든 문제를 끝낼 때까지 빈칸 맞추기를 한다. 학점이 F가 되거나 모든 문제를 마치면 루프를 종료하고 메인 메뉴로 이동한다.

예정 : 현재 하나의 프리셋 텍스트만을 다룰 수 있다. 다른 클래스들과 결합하여 무작위 텍스트에 대해 진행할 수 있도록 개발한다. 그리고 레이아웃을 다듬어 클래스의 완성도를 높인다.



6) Statistics | 통계

개요 : 저장된 사용자의 타자 속도, 타자 정확도, 게임 승률 등의 **통계를 출력**한다.

완료 : TypingManager와 세 개의 파생 클래스, 미니 게임 클래스 등 모든 기능 클래스가 완성되면 개발할 예정이다. 따라서 아주 기본적인 틀만 구현하였다.

예정 : 모든 클래스가 응용 가능한 수준까지 개발된 후 구현한다. 기본적인 흐름 설계는 다음과 같다. 첫째, 사용자가 원하는 기능(단어, 게임 등)을 선택한다. 둘째, 해당 기능에서의 타자 속도, 타자 정확도, 승률 등의 통계를 제공한다.

라. 테스트 (프로토타입)

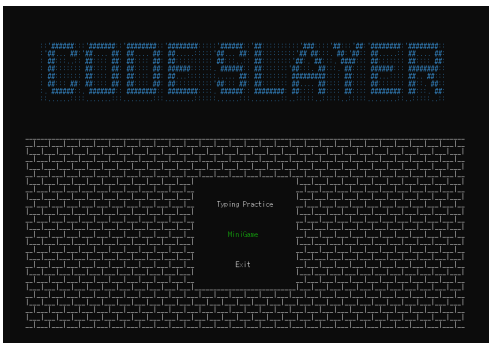
현재까지 **개발한 내용을 테스트**한다. 분류 항목은 좌측 그림의 실행 위치를 나타낸다. 내용 항목에서는 좌측 그림에서 어떤 클래스가 응용되었고 테스트 되었는지를 상세하게 서술한다.

*** ProgramManager ***

분류 : ProgramManager 클래스

내용 :

- ProgramManager : 프로그램의 전반적인 흐름 관리
- ProgramManager : 프로그램에 필요한 객체 생성
- ProgramManager : 소개 화면 출력
- ProgramManager : 사용자가 호출한 기능 실행
- Console : 출력 좌표 및 색상 설정
- Console : 텍스트 파일을 읽어 설정한 좌표와 색상으로 출력
- Config : 열거형 스크린 클래스 사용
- Keyboard : 엔터를 입력하여 메인 메뉴로 이동



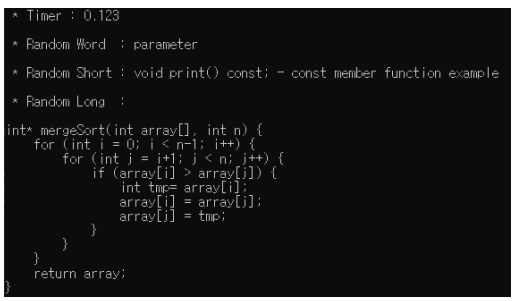
*** MainMenu ***

분류 : MainMenu 클래스

내용 :

- MainMenu : 메인 메뉴 관리
- MainMenu : 사용자의 키보드 입력을 받아 각 항목 이동
- MainMenu : ProgramManager로 다음 기능의 스크린 반환
- Config : 열거형 스크린 클래스 사용
- Console : 출력 좌표 설정
- Console : 출력 색상 설정
- Console : 텍스트 파일을 읽어 설정한 좌표와 색상으로 출력
- Keyboard : 화살표와 엔터를 입력하여 각 항목 이동

마. 테스트 (기능)

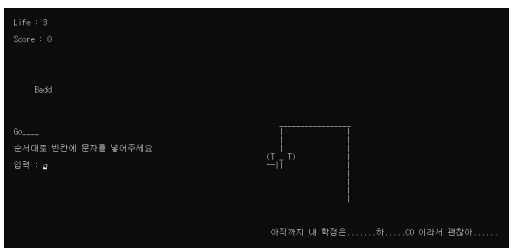


*** Timer, Text, TypingManager***

분류 : #TestSite

내용 :

- Timer : 시간 측정
- Text : 문자열 저장
- Text : 문자열 반환
- TypingManager : 모든 텍스트 파일을 Text 벡터에 저장
- TypingManager : 무작위 단어 불러오기
- TypingManager : 무작위 짧은 글 불러오기
- TypingManager : 무작위 긴 글 불러오기



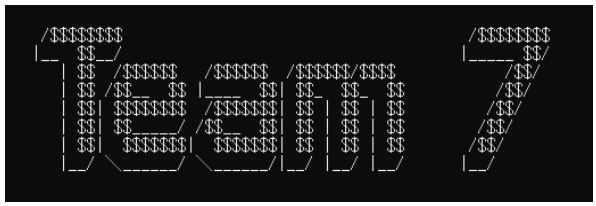
*** MiniGame ***

분류 : #TestSite

내용 :

- MiniGame : 미니 게임 메인 루프 실행
- MiniGame : 레이아웃 및 출력
- MiniGame : 사용자 키보드 입력
- MiniGame : 정답 판별 등 게임 논리 확인

5. 팀원 구성



*** 팀 세븐 ***

가. 팀 소개

1) 팀명 : Team 7
2) 팀장 : 윤건호
3) 팀원 : 김상원, 안치원, 우은혁

나. 역할 분담

1) 팀장 윤건호

(1) 프로젝트 지휘
(2) 프로젝트 관리
(3) 프로젝트 문서화
(4) 클래스 구현
- 주 : ProgramManager, MainMenu, Statistics
- 보조 : Config, Console, Random, Keyboard(추가)

(5) 시나리오 테스트
- 오류 수정
- 성능 개선

윤건호	
번호	역할
1	프로젝트 지휘
2	프로젝트 관리
3	프로젝트 문서화
4	클래스 구현
5	시나리오 테스트

2) 팀원 김상원

(1) 클래스 구현
- 주 : MiniGame

(2) 데이터베이스 추가
- 미니 게임 프리셋 데이터 추가
- 타자 연습 프리셋 데이터 추가

(3) 시나리오 테스트
- 오류 수정
- 성능 개선

김상원	
번호	역할
1	클래스 구현
2	데이터베이스 추가
3	시나리오 테스트

3) 팀원 안치원

(1) 클래스 구현
- 주 : TypingManager, WordTyping, ShortTyping, LongTyping
- 보조 : Timer, Text

(2) 데이터베이스 추가
- 타자 연습 프리셋 데이터 추가
- 미니 게임 프리셋 데이터 추가

(3) 시나리오 테스트
- 오류 수정
- 성능 개선

안치원	
번호	역할
1	클래스 구현
2	데이터베이스 추가
3	시나리오 테스트

4) 팀원 우은혁

(1) 클래스 구현
- 주 : WordTyping, ShortTyping, LongTyping, Statistics
- 주 : TypingManager(추가)

(2) 데이터베이스 추가
- 타자 연습 프리셋 데이터 추가
- 미니 게임 프리셋 데이터 추가

(3) 시나리오 테스트
- 오류 수정
- 성능 개선

우은혁	
번호	역할
1	클래스 구현
2	데이터베이스 추가
3	시나리오 테스트

6. 개발 일정

가. 개요

제안서 작성일로부터 6월 7일까지의 **CodeSlayer 개발 일정**을 나타낸다. 중간보고서 작성 시점인 5월 27일을 기준으로 작성하였다. 개발이 끝난 전체 일정은 회색으로 표시하였다. 팀원들은 각자 주요 담당 클래스를 설정하였고, 맡은 클래스를 책임지며 중점적으로 개발한다. 다른 팀원들의 진행 정도에 따라 자신이 담당하는 클래스를 지속해서 수정 및 개선한다.

가장 먼저 표의 형태로 전체적인 프로젝트 일정을 제시한다. 그리고 개인 일정을 제시한다. 개인 일정은 각자 맡은 역할이 명확할 때는 자세하게 구분하였으나, 공통 작업의 경우에는 개인 일정 없이 전체 일정에만 작성하였다.

나. 전체 일정

구분	개발 세부 내용	5.7 ~ 5.13	5.14 ~ 5.21	5.21 ~ 5.28	5.29 ~ 6.4	6.5 ~ 6.7
주제 선정	주제 창출					
	주제 선정					
	아이디어 창출					
	아이디어 선정					
준비	유사 프로그램 분석					
	기초자료 수집					
	제안서 작성					
설계	클래스 구조 설계					
	기능 설계					
	디자인 설계					
개발	보조 클래스 개발					
	주 클래스 개발					
	클래스 간 연결					
	통합 개발					
개선	오류 수정					
	데이터 추가					
테스트	시나리오 테스트					
	디버깅					
최종 준비	최종 문서 작업					
	프로젝트 공개					
보수	데이터베이스 업데이트					

*** 개발 일정표 ***

다. 개인 일정

다음 두 페이지는 **팀원들의 대략적인 개인 일정**을 나타낸다. 제안서 작성 시점부터 6월 7일까지의 일정을 크게 세 부분, 5월 넷째 주, 5월 다섯째 주, 6월 첫째 주로 나누었다. 중간보고서 작성 시점인 5월 27일을 기준으로 작성하였으며 개발이 끝난 개인 일정은 밑줄로 표시하였다.

프로그램 구현 과정에서 서로 긴밀하게 협력해야 해결할 수 있는 작업이 많을 것으로 예상하였다. 따라서 공통 사항으로 처리한 일정들이 있으며, 이러한 공통 일정은 개인 일정에 표기하지 않았다. 나타내지 않은 모든 일정은 협력하여 처리한다.

윤건호

지난 일정
개발 완료

상세

01 5월 4주차
클래스 구현

02 5월 5주차
클래스 개발, 통합 개발

03 6월 1주차
시나리오 테스트, 디버깅

클래스 구현
1. (90%) 보조 클래스 Config, Console, Random 개발
2. (90%) 보조 클래스 Keyboard 개발
3. (90%) 주 클래스 ProgramManager, MainMenu 기본 틀 구현

클래스 개발, 통합 개발
1. (80%) 주 클래스 ProgramManager, MainMenu 개발
2. (10%) 주 클래스 Statistics 개발
3. (30%) 통합 개발

시나리오 테스트
1. 프로젝트 시나리오 테스트
2. 오류 수정 및 코드 개선

김상원

지난 일정
개발 완료

상세

01 5월 4주차
클래스 구현

02 5월 5주차
클래스 개발, 데이터 추가

03 6월 1주차
시나리오 테스트 , 디버깅

클래스 구현
1. (80%) 주 클래스 MiniGame 기본 틀 구현

클래스 개발, 데이터 추가
1. (60%) 주 클래스 MiniGame 개발
2. (90%) 프로그램에서 사용하는 프리셋 데이터 추가

시나리오 테스트
1. 주 클래스 MiniGame 시나리오 테스트
2. 오류 수정 및 코드 개선

안치원

지난 일정
개발 완료

상세

01 5월 4주차
클래스 구현

02 5월 5주차
클래스 개발, 데이터 추가

03 6월 1주차
시나리오 테스트 , 디버깅

클래스 구현
1. **(90%) 보조 클래스 Timer, Text 기본 틀 구현**
2. **(80%) 주 클래스 TypingManager 기본 틀 구현**

클래스 개발, 데이터 추가
1. (50%) 주 클래스 TypingManager 개발
2. (10%) 주 클래스 WordPractice, ShortPractice, LongPractice 개발
3. **(90%) 프로그램에서 사용하는 프리셋 데이터 추가**

시나리오 테스트
1. 작성한 클래스의 시나리오 테스트
2. 오류 수정 및 코드 개선

우은혁

지난 일정
개발 완료

상세

01 5월 4주차
클래스 구현

02 5월 5주차
클래스 개발, 데이터 추가

03 6월 1주차
시나리오 테스트 , 디버깅

클래스 구현
1. (10%) 주 클래스 Statistics 개발
2. **(80%) 주 클래스 TypingManager 기본 틀 구현**

클래스 개발, 데이터 추가
1. (50%) 주 클래스 TypingManager 개발
2. (10%) 주 클래스 WordPractice, ShortPractice, LongPractice 개발
3. **(90%) 프로그램에서 사용하는 프리셋 데이터 추가**

시나리오 테스트
1. 작성한 클래스의 시나리오 테스트
2. 오류 수정 및 코드 개선

7. 기대 효과

가. 현주소

컴퓨터와 스마트폰이 널리 보급된 오늘날, 한글 타자가 느린 것을 불평하는 사람은 거의 없다. 하지만 영어 타자와 코딩은 그렇지 않다. 자주 사용할 일이 없기에, 한글 타자 속도와 수 배까지 차이가 나고는 한다. 따라서 우리에게 **친숙하지 않은 영어와 코드**로 이루어진 C++ 언어를 작성하기는 쉽지 않다.

```
////////////////////////////////
/* Change Color Name to Integer */
////////////////////////////////
int Console::ColorNameToNumber(const string& colorName) const
{
    if (colorName == "default")
        return -1;

    if (colorName == "random")
    {
        int colorSet[5] = { 12,10,3,13,14 };
        return colorSet[rand() % 5];
    }
    else if (colorName == "red")    return 12;  // RED
    else if (colorName == "green")  return 10;  // GREEN
    else if (colorName == "blue")   return 3;   // BLUE
    else if (colorName == "purple") return 13;  // PURPLE
    else if (colorName == "yellow") return 14;  // YELLOW
    else if (colorName == "white")  return 15;  // WHITE
    else if (colorName == "black")  retunr
```

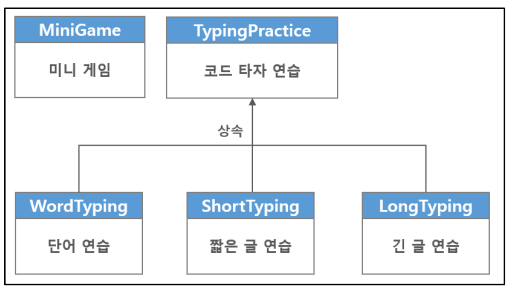
* 잦은 오타 *

Result	Score	Code
Wrong Answer	0 /100	C
Wrong Answer	0 /100	C
Output Limit	0 /100	C
Compile Error	0 /100	C
Run-Time Error	0 /100	C
Compile Error	0 /100	C
Run-Time Error	0 /100	C
Run-Time Error	0 /100	C

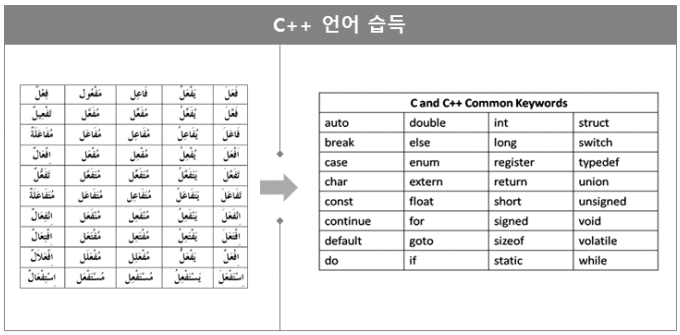
* 부족한 시험 시간 *

나. CodeSlayer 개발 후 기대 효과

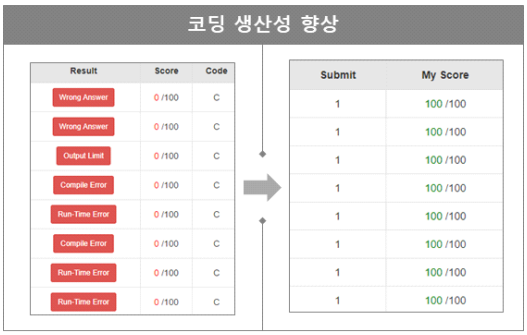
CodeSlayer는 C++ 언어와 타자 연습, 그리고 게임을 결합한 프로그램이다. 크게 두 가지 주요 기능을 제공한다. 첫째, 코드 타자 연습 기능이다. 이를 활용하여 사용자의 **타자 실력(정확도, 속도)을 증진**한다. 둘째, 미니 게임 기능이다. 이를 활용하여 사용자가 **코딩에 흥미**를 느끼게 한다. CodeSlayer가 제공하는 두 가지 기능을 적절하게 활용한다면, 사용자는 C++ 언어에 더 익숙해질 것이다. 이는 결과적으로 사용자의 **코딩 생산성 향상**으로 이어진다.



* CodeSlayer가 제공하는 주요 기능 활용 *



* C++ 언어 체화 *



* 코딩 생산성 향상 *