본 PSet 은 저의 강의 경험과 학생들의 의견 및 Stanford CS106 과 Harvard CS50 같은 강의에서 수집된 자료를 토대로 작성되었습니다. 본 PSet 에 문제가 있거나, 질문 혹은 의견이 있다면, 언제든지 알려 주시면 감사하겠습니다. 강의 개선에 많은 도움이 되겠습니다.
idebtor@gmail.com

Problem Set - HelloWho

목차

Getting Started

이 첫 번째 PSet 에서 각자의 컴퓨터에 프로그래밍 환경을 설정하고 Piazza 에 가입할 것입니다. 또한, 콘솔에서 입력을 받아 요청한 대로 입력을 처리하는 첫 번째 프로그램을 작성할 것입니다.

GitHub 에서 https://github.com/idebtor/nowic 리포지토리를 본인의 컴퓨터로 가져오세요. 이 리포지토리를 **"읽기 전용"으로 유지하세요**. 이 리포지토리를 본인이 쉽게 접근할 수 있는 본인의 리포지토리(예를 들면, nowicx) 혹은 개발 폴더에 복사하세요. 다음 형식과 같아야 합니다:

```
~/nowicx/psets/pset1/pset1.pdf            # this file
~/nowicx/psets/pset1/hellofunc.exe        # a solution to compare your work for Windows
~/nowicx/psets/pset1/hellofunc            # a solution to compare your work for macOS
~/nowicx/psets/pset1/names.txt            # a list of names used in Step 2.
```

개요: Hello <Who>!

이 PSet 에서 배울 것은

- 명령줄에서 g++ 컴파일 및 실행하기
- 줄바꿈 문자 '₩n'과 'endl' 사용하기
- **main(int argc, char *argv[])**으로 전달되는 명령줄 인수 처리하기
- **iostream, cout, cin, getline()**을 사용하여 콘솔에서 i/o 처리하기
- 함수와 프로토타입 사용하기

- 여러 소스 파일 컴파일 및 링크하기
- Piping

Step 1. Create a source file: hellowho.cpp

아래와 같이 **~/nowicx/pset1/hellowho.cpp** 에 본인의 소스 파일을 작성하세요.

```
/*
* file: hellowho.cpp
* It prints "Hello World!" or "Hello" with a given name.
* The completed code should work as shown below. “>” is a prompt of the console.
*
* To run the program without a command line argument:
*    > ./hello
*    > Enter a name: John Lee
*    > Hello John Lee!
*    > Enter a name: Peter Kim
*    > Hello Peter Kim!
*    > Enter a name:<Enter>
*    > Hello World!
*    >
*
* To run the program with a command line argument:
*    > ./hello John "Dr. Lee" "Handong Global University" peter
*    > Hello John!
*    > Hello Dr. Lee!
*    > Hello Handong Global University!
*    > Hello peter!
*    > Hello World!
*    >
* To run the program through a pipe
* (names.txt contains a list of names as shown below:)
*    > cat names.txt | ./hello
*    > Enter a name: Hello john!
*    > Enter a name: Hello Dr. Lee!
*    > Enter a name: Hello Handong Global University!
*    > Enter a name: Hello Peter!
*    > Enter a name: Hello World!
*
*    2020/02/10: Created, idebtor@gmail.com
*    2020/12/15: working in vs code
*/

#include <iostream>
#include <string>
using namespace std;

int main() {
  // Use setvbuf() to prevent the output from buffered on console.
  // setvbuf(stdout, NULL, _IONBF, 0);

  cout << "Your code here" << endl;
  cout << "Hello World!\n";

  // Use system("pause") to prevent the terminal from disappearing
  // as soon as the program terminates as in Visual Studio sometimes.
  // system("pause");
  return EXIT_SUCCESS;
}
```

hellowho.cpp 는 결과적으로 파일 맨 위의 주석에 나온 것처럼 작동해야 합니다. 이 프로그램은 명령줄 인수의 존재 여부에 따라 다르게 작동합니다. 자기 자신을 실행하는 파일(또는 hellowho.exe) 외에 다른 명령줄 인수가 없다면 사용자에게 이름을 입력하도록 요청합니다. 사용자가 이름을 입력하면 이름과 함께 “Hello”를 출력합니다. 그렇지 않으면 “Hello World!”를 출력하고 프로그램을 종료합니다. 주어진 인수가 있다면 이름이 주어졌다고 가정합니다. loop 문을 사용하여 주어진 인수를 모두 출력하고 “Hello World!”를 출력하며 프로그램을 종료하도록 코딩하세요.

Hint 1. 사용자 이름 입력받기

강의 노트 "getinputs.md"를 **반드시** 복습하세요. 노트는 다음 경로에서 확인할 수 있습니다:

nowic/pset1/**getinputs.md**

강의에서 말씀드린 cin 이 아닌 getline()을 사용하는 것이 핵심입니다. 사용자가 명령줄에서 이름을 입력하지 않을 경우, 사용자가 엔터를 입력하거나 아무것도 입력하지 않을 때까지 상호적이며 반복적으로 이름을 입력할 수 있어야 합니다. 사용자가 <Enter>를 입력하면 “Hello World”를 출력하고 프로그램을 종료합니다. 실행 예시는 아래와 같습니다.

실행 예시:

```
PS C:\GitHub\nowicx\psets\pset01hellowho> g++ hellowho.cpp -o hellowho
PS C:\GitHub\nowicx\psets\pset01hellowho> ./hellowho
Enter a name: john
Hello john!
Enter a name: Dr. Lee
Hello Dr. Lee!
Enter a name:
Hello World!
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

Hint 2. 컴파일 및 링크하기

컴파일은 소스 코드 파일(.c, .cc, 또는 .cpp)을 처리하고 ‘object’ 파일을 생성하는 것을 얘기합니다. 이 단계에서 사용자가 실제로 실행할 수 있는 무언가를 생성하지는 않습니다. 컴파일러는 단지 컴파일된 소스 코드 파일에 해당하는 기계 언어 명령만 생성합니다. 다음 명령줄은 hellowho.o 를 생성합니다:

```
g++ -c hellowho.cpp
```

링크는 여러 객체 파일에서 하나의 실행 파일을 생성하는 것을 얘기합니다. 이 단계에서 보통 링커는 정의되지 않은 함수(일반적으로 main 자체)를 잡아냅니다. 컴파일 과정에서 컴파일러가 특정 함수의 정의를 찾을 수 없는 경우, 함수가 다른 파일에 정의되어 있다고 가정합니다.

다음 명령줄은 필요한 모든 객체 파일을 링크하고 실행파일(PC 는 hellowho.exe 또는 macOS 는 hellowho)을 생성합니다.

```
g++ hellowho.o -o hellowho
```

파일이 많지 않은 경우 아래와 같이 하나의 명령줄에서 컴파일과 링크를 수행할 수 있습니다.

```
g++ hellowho.cpp -o hellowho
```

Step 2: 명령줄 인수 사용하기

명령줄 인수에 대한 강의 노트 복습을 권장합니다. 강의 노트는 **nowic/pset1/argcargv.md.**에서 확인할 수 있습니다.

다음 명령을 사용해서 **hellowho.cpp** 의 복사본 **helloargs.cpp** 를 생성하세요.

```
$ cp hellowho.cpp helloargs.cpp
```

이제 **명령줄**을 활용하여 이름 목록을 전달하면 프로그램이 각 이름을 처리하도록 할 것입니다. 프로그램에서 이 부분은 명령줄 인수를 받아야 합니다. 그런 다음, 아래와 같이 **helloargs.cpp** 에 **main()**을 선언해야 합니다.

```
int main(int argc, char *argv[])
```

첫 번째 인수 **argc** 는 명령줄 인수의 개수를 가집니다. 예를 들어, 명령줄이 다음과 같다면

```
./helloargs john "Dr. Lee" handong peter
```

argc 와 **argv** 는 시스템에 의해 자동으로 다음과 같이 설정됩니다.

```
argc = 4
argv[0] = "C:/GitHub/nowicx/psets/pset1/helloargs"
argv[1] = "john"
argv[2] = "Dr. Lee"
argv[3] = "handong"
argv[4] = "peter"
```

argv 는 문자열의 "배열"입니다. 배열은 줄지어 있는 헬스장 사물함이라고 생각할 수 있습니다. 각 사물함 안에 (양말 같은 ^^) 어떤 물건이 들어있고, 이 사물함 안이 곧 문자열입니다. 첫 번째 사물함을 열기 위해(i.e., "index into"), 배열의 index 는 0 부터 시작하므로 **argv[0]**과 같은 구문을 사용합니다. 다음 사물함을 열기 위해, **argv[1]**과 같은 구문을 사용합니다. 이어서 같은 방법으로 접근합니다. **n** 개의 사물함이 있다면, **argv[n]**번째 사물함은 존재하지 않으므로 (혹은 그 사물함이 존재해도 다른 사람의 사물함이라고 보면 됩니다. 어찌 됐든 그 사물함을 열어서는 안 되므로) **argv[n - 1]**번째 사물함까지 열 수 있습니다. 즉, **argv** 가 문자열의 배열이듯이, **문자열**도 문자의 배열입니다. 따라서 대괄호를 사용하여 **argv** 의 개별 문자열에 접근한 것처럼 문자열의 개별 문자에 접근할 수 있습니다.

인수가 두 개 이상의 단어로 구성된 경우 큰따옴표를 입력해야 합니다. 다음과 같이 명령줄 인수를 처리할 수 있어야 합니다:

실행 예시: (주어진 명령줄 인수가 없더라도 동일한 결과를 출력합니다.)

```
PS C:\GitHub\nowicx\psets\pset01hellowho> ./helloargs john "Mr. Lee" "Handong Global" peter
Hello john!
Hello Mr. Lee!
Hello Handong Global!
Hello peter!
Hello World!
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

```
PS C:\GitHub\nowicx\psets\pset01hellowho> ./helloargs
Enter a name: Mr. Lee
Hello Mr. Lee!
Enter a name:
Hello World!
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

Step 3: 여러 파일을 사용해서 실행 파일 생성하기

이 step 에서는 아래와 같이 명령줄 인수의 이름을 **printfunc.cpp** 라는 별도의 파일에 정의된 함수 **printfunc()**으로 전달하고자 합니다:

```
// file: printfunc.cpp
// C++ for C Coders & Data Structures
// Lecture note by idebtor@gmail.com
#include <iostream>

void printfunc(int n, char *names[]) {

    for (int i = 0; i < n; i++)
     std::cout << "Hello " << names[i] << "!" << std::endl;

}
```

위와 같이 **printfunc.cpp** 파일을 작성하세요.

다음 명령을 사용해서 **helloargs.cpp** 의 복사본 **hellofunc.cpp** 를 생성하세요.

```
$ cp helloargs.cpp hellofunc.cpp
```

- **hellofunc.cpp** 의 출력이 이전 단계의 출력과 같도록 파일을 수정하세요. **hellofunc.cpp** 에서 **printfunc()**을 사용하기 위해서 먼저 hellofunc.cpp 에 함수 프로토타입을 정의해야 합니다.
- 명령줄 인수에서 사용자의 입력을 받을 경우 **printfunc()**을 사용하세요.

실행 예시: (주어진 명령줄 인수가 없더라도 동일한 결과를 출력합니다.)

```
PS C:\GitHub\nowicx\psets\pset01hellowho> ./hellofunc john "Mr. Lee" "Handong Global" peter
Hello john!
Hello Mr. Lee!
Hello Handong Global!
Hello peter!
Hello World!
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

```
PS C:\GitHub\nowicx\psets\pset01hellowho> g++ hellofunc.cpp printfunc.cpp -o hellofunc
PS C:\GitHub\nowicx\psets\pset01hellowho> ./hellofunc
Enter a name: Peter Kim
Hello Peter Kim!
Enter a name:
Hello World!
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

Hint 3. 여러 파일을 컴파일 및 링크하기

프로그램이 복잡해짐에 따라 소스 코드를 별도의 파일에 저장하고자 합니다. 예를 들어, 'show()'라는 함수가 'show.cpp'에 정의되어 있고 아래와 같이 'show()'에서 호출된다고 가정해 봅시다:

```
// file: show.cpp
#include <iostream>
void show(int a) {
    std::cout << a << std::endl;
}
```

프로그램이 논리적으로 나뉘어 작성될 수 있도록 C++은 **분할 컴파일**을 지원합니다. 프로그램을 여러 파일로 분할할 수 있으며, 각 파일을 독립적으로 컴파일할 수 있습니다. 'show.cpp'에는 'main()'이 없으므로 **컴파일러가 실행 파일을 생성할 수 없지만,** '.exe' 대신 확장명이 '.o'인 **객체** 파일을 생성할 수 있습니다.

```
$ g++ -c show.cpp     # compilation: produces show.o
```

다음 파일 'hello.cpp'로 실행 파일을 생성하려면 컴파일러에게 사용되는 모든 코드를 어디에서 찾아야 하는지 알려줘야 합니다.

```
#include <iostream>
void show(int a);   // lets the compiler know it from external sources
                    // this is called a function prototype
                    // the header file contains a list of function prototypes
int main() {
    show(123);
}
```

이 파일들을 다음과 같이 컴파일할 수 있습니다.

```
$ g++ hello.cpp show.cpp -o hello      # compile & link: produces hello.exe
$ ./hello                              # runs hello.exe
```

이제 'hello' 프로그램을 실행할 수 있습니다.

소스 파일 중 하나를 변경한 경우, 실제로 변경된 파일만 다시 컴파일하려고 합니다. 각 파일을 분할해서 컴파일할 수 있는 방법이 필요합니다. 이 과정에서 보통 파일에 **객체 코드**가 포함되어 있음을 알려주는 .o 또는 .obj 파일 확장자를 가진 파일을 생성합니다.

컴파일러는 객체 파일을 서로 **링크**하여 실행 파일을 생성합니다. 시스템(Windows & gcc)에서는 다음과 같이 프로그램을 분할 컴파일합니다:

```
$ g++ -c show.cpp                  # compilation: produces show.o
$ g++ -c hello.cpp                 # compilation: produces hello.o
$ g++ show.cpp hello.cpp           # linking: produces a.exe
$ g++ show.cpp hello.cpp -o hello  # linking: produces hello.exe
```

추가 GCC 컴파일러 옵션

컴파일러 옵션에 사용할 수 있는 몇 가지 플래그가 있습니다.

```
$ g++ -std=c++11 -Wall file1.cpp file2.cpp -o prog   # generates prog.exe
```

- -o: 출력 실행 파일 이름을 지정합니다.
- -std=c++11: C++ standard(c++11, c++14, c++17, c++2a)를 명시적으로 지정합니다.
- -Wall: 대부분의 경고 메시지를 활성화합니다.

GCC 및 Make 를 위한 좋은 가이드라인 :
[Compiling, Linking and Building C/C++ Applications]

Step 4: 'Pipe' 사용하기

이 부분은 **'pipe'**의 개념을 소개하는 부분입니다. 프로그램을 수정하지 않고도 원활하게 실행되어야 합니다. 리디렉션(redirection)을 통해 pipe 프로세스의 출력에서 입력(또는 이름)을 받아옵니다. Mac 과 Linux 에서는 **cat** 대신 **type** 을 사용하세요. 이 부분이 제대로 작동하지 않는다면 아래와 같이 작동하도록 코드를 수정하세요:

실행 예시:

```
PS C:\GitHub\nowicx\psets\pset01hellowho> cat names.txt
john
Dr. Lee
Handong Global University
Peter
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

```
PS C:\GitHub\nowicx\psets\pset01hellowho> cat names.txt | ./hellofunc
Enter a name: Hello john!
Enter a name: Hello Dr. Lee!
Enter a name: Hello Handong Global University!
Enter a name: Hello Peter!
Enter a name: Hello World!
PS C:\GitHub\nowicx\psets\pset01hellowho>
```

과제 제출

- 소스 파일 상단에 아래와 같이 아너 코드 문장을 적고 서명하세요.
  On my honor, I pledge that I have neither received nor provided improper assistance in the completion of this assignment.
  서명: _____ 학번: _____
- 제출하기 전에 코드가 제대로 컴파일이 되고 실행되는지 확인하세요. 제출 직전에 급하게 코드를 수정한 후 코드가 제대로 컴파일이 될 거라고 짐작하지 않는 게 좋습니다. "거의" 작동하는 코드도 틀린 것입니다.
- 과제가 컴파일 및 실행된다면, 마감 기한 전까지 과제의 일부만 완성했더라도 제출하기 바랍니다. 컴파일 및 실행되지 않는다면 제출하지 마세요. 마감 시간 이후 24 시간 이내 제출하면, 만점에서 25% 감점하고 채점합니다. 그 이상 늦은 것은 채점하지 않으며, 0 점 처리합니다.
- 제출 후, **마감 기한 전까지** 수정 및 재제출이 가능합니다. 파일 하나만 수정하더라도 해당 파일과 관련된 파일들을 모두 재제출해야 합니다. 재제출 횟수는 제한 없습니다. 마감 기한 전에 **가장 마지막으로** 제출된 파일을 채점할 것입니다.

제출 파일 목록

- 본인의 소스 파일을 기한 내에 Piazza **pset1 폴더**에 제출하세요.
  - hellowho.cpp, helloargs.cpp, hellofunc.cpp
- 두 분반의 학생들 모두 같은 폴더를 사용하므로 **TA 의 가이드라인에 따라** 파일을 제출해 주세요. 가이드라인을 따르지 않을 경우 불이익이 있습니다. 파일 제출 시 제출 시간이 표기된다는 것을 기억하세요.