**Задание 3. Запросы с аргументами**

**Что нужно сделать**

Сделайте запрос к сервису httpbin.org, отправив при этом несколько дополнительных аргументов с данными.

Все названия аргументов и их значения строковые и принимаются от пользователя. Пользователь последовательно вводит их названия вместе со значениями, пока не встретится аргумент с названием “post” или “get”, что будет означать, что ввод аргументов закончен и требуется отправить их на сервер выбранным способом.

Если пользователь выбрал “post”, то выполняется POST-запрос и аргументы отправляются POST-способом (как форма). Если “get”, то выполняется GET-запрос с GET-аргументами (прямо в URL).

По результатам выполнения запроса выведите ответ сервера в стандартную консоль и обратите внимание на поля form или args, в зависимости от типа запроса в которых должны быть перечислены все введённые вами аргументы.

**Рекомендации**

Чтобы создать экземпляр объекта cpr::Payload для POST-запроса, используйте конструктор, который принимает диапазон итераторов массива от begin() до end(), соответственно. При этом тип элементов массива должен быть cpr:Pair, а сам массив содержит в себе имена параметров вместе с их значениями. Каждый элемент можно инициализировать с помощью конструктора cpr::Pair((std::string)key, (std::string)value). Обратите внимание на использование эксплицитного приведения к строкам. Это может понадобиться, если используется старая версия библиотеки cpr.

Согласно HTTP-стандарту, аргументы в GET-запросах перечисляются прямо в URL запроса после вопросительного знака. Названия аргументов отделяются от их значений с помощью символа “=”, а сами аргументы разделены символами “&”. Пример аргументов, заданных в URL запроса:

?foo=first&animal=cat&bar=third

Для хранения названий параметров и их значений используйте словарь:

std::map<std::string, std::string> arguments;

В таких словарях могут содержаться данные вида {“height”, “185”}, {“weight”, “75”}. В случае с нашими запросами там будет что-то другое, но идея та же: название — значение.  Чтобы пройти по такому словарю, потребуется итератор вида std::map<std::string, std::string>::iterator it; Используйте его так:

for (it = arguments.begin(); it != arguments.end(); it++)  
{  
    it->first *// даёт доступ к названию параметра*  
*it->second // даёт доступ к значению параметра*  
*}*

**Что оценивается**

Корректность работы программы при различных пользовательских данных.