ДЗ

По теме Полиморфизм, Шаблонное программирование

ххх

ххх

Создать 2 объекта:

Объекты:

1. Язык программирования C
2. Язык программирования C++

Язык C должен поддерживать только базовые арифметические операции:

1. Функции сложения, вычитания, умножения, деления
2. Функции взятия адреса и разыменования

Язык C++ умеет всё, что умеет язык C, а также поддерживает интерфейс создание классов

объект языка C:

Содержит в себе:

1. map<VARIABLE_NAME, VARIABLE_VALUE>
   Для упрощения класс может содержать только переменные типа **int**
2. И функции для работы с этими переменными
   a. Увеличение/Уменьшение на/в *какое-то кол-во раз*, но не изменяя сами переменные, просто выводить их измеренные значения
   b. Взятие адреса переменной, а та же , получение переменной по адресу

объект языка C++:

Содержит в себе:

1. Всё тоже самое, что язык объект C, но все данные можно упаковать в class

Пример кода, для упаковки значений в class доступе по следующей ссылке

Но не запрещенно упрощать/изменять этот момент до любого комфортно и понятно для вас вида, но прикрепленный код уже полность рабочий

Взаимодействия, отношения, наследования и имплементацию всех структур описать самостоятельно!

```cpp
#include <iostream>

using namespace std;

template<char VARIABLE_NAME, int VARIABLE_VALUE>
struct getPair {
    static char const   variable_name   = VARIABLE_NAME;
    static int  const   variable_value  = VARIABLE_VALUE;

};


template<typename... VARIABLE>
struct  tail_class;

template<typename FIRST_PAIR, typename... OTHER_PAIR>
struct  tail_class<FIRST_PAIR, OTHER_PAIR...> {
    static char const   variable_name   = FIRST_PAIR::variable_name;
    static int  const   value           = FIRST_PAIR::variable_value;
    using               next            = tail_class<OTHER_PAIR...>;
    static int getVariable(char variable) {
        if (variable == variable_name) {
            return value;
        }
        else {
            return next::getVariable(variable);
        }
    }
};

template<>
struct  tail_class<> {
    static int getVariable(char variable) {
        return NULL;
    }
};

template<char NAME_CLASS, typename CLASS_TAIL>
struct _class {
    static char const   class_name = NAME_CLASS;
    using               tail_class = CLASS_TAIL;
    static int getVariable(char variable) {
        return tail_class::getVariable(variable);
    }
};

template<typename... CLASSES>
struct listClass {

};


int main()
{


    //Все значения в следующем коде обязаны быть const значения
    //-----------------------------------
    _class<'a', tail_class  <
        getPair<'i', 0>,
        getPair<'g', 10>,
        getPair<'p', 3>
    >
    > a_igp;
    //-----------------------------------
    //Всё, остальные не обязаны, в данном коде создался класс с именем 'a', и перемерными {i,
    char name_variable;
    cin >> name_variable;
    cout << a_igp.getVariable(name_variable);

}
```

Подсветка програмного кода