template template template

فرض کنید یک کلاس Template به نام Storage که در پایین implement شده است داریم.

```
template <class T>
class Storage
{
private:
    T m_value;
public:
    Storage(T value)
    {
        m_value = value;
    }

    ~Storage()
    {
    }

    void print()
    {
        std::cout << m_value << '\n';
    }
};
```

همانطوری که مشخص است این کلاس کارش ذخیره سازی داده از نوع های متفاوت می باشد. حال سوال اینجاست آیا واقعا برای تمام انواع متغیر به صورت درست عمل می کند؟ فرض کنید این کلاس را بخواهیم برای نوع pointer ای استفاده کنیم آیا همچنان اگر تابع print را صدا بزنیم به صورت درست کار می کند؟ اگر درست کار نمی کند پیشنهاد شما برای درست کردنش چیست؟ (به عنوان مثال اگر بخواهیم برای int* استفاده کنیم. منظور از int* آرایه ای از integer ها نیست و تنها یه پوینتر مدنظرمون هستش به عنوان مثال کد زیر را در نظر بگیرین) دقت کنید فقط از نوع int* نیست و انواع پوینتر های مختلف مثل float* و ... نیز مدنظرمون می باشد.

```
int main()
{
```

```
    Storage<int> myint(5);
    myint.print();


    int x = 7;
    Storage<int*> myintptr(&x);


    x = 9;
    myintptr.print();

    return 0;
}
```

انتظار داریم پس از اجرای کد بالا عدد های 5 و 7 چاپ شوند.

حال اگر بخواهیم برای نوع متغیر char* که درواقع آرایه ای char ها می باشد از این کلاس استفاده کنیم باید چه کنیم؟

```
int main()
{

    char* string = new char[40];

    std::cout << "Enter your name: ";
    std::cin >> string;

    Storage<char*> storage(string);

    delete[] string;

    storage.print();
}
```

پس در نهایت کدتون به عنوان مثال باید برای همچین main ای کار بکند.

```
int main()
{
```

```
    Storage<int> myint(5);
    myint.print();

    double x = 7.5;
    Storage<double*> myintptr(&x);

    x = 9;
    myintptr.print();

    char *name = new char[40];
    strcpy(name, "Alex");

    Storage< char*> myname(name);

    delete[] name;

    myname.print();
}
```