می توان typename ها را در کلاس قالب به صورت یک جفت (pair) نگه داشت :

Template<typename U , typename V>

به جای typename می توان از لغت class استفاده کرد

Template ها را می توان برای function ها نیز استفاده کرد

Template<typename T>

Void show(T arr[] , int n ) {

For(int i = 0 ; i < n ; i++)

Cout << arr[i] << ‘ ‘;

}

اما نکته ای که برای توابع وجود دارد این است که موقع صدا کردن بعد اسم تابع نیاز به مشخص کردن نوع پارامتر ورودی نیست ( یعنی احتیاجی به show<vector<int>>(v,n) نیست )

از template class ها می توان ارث بری داشت به اشکال زیر :

Class MyLyst : public List<int> {….}

Template <typename T> class MyList : public List<T> {…}

Template <typename T> class MyList : public List<int> {…}

نکته مهم این است که حتی در ارث بری برای کلاس های template نوع type باید مشخص شود

حال چه با type name چه با type های مشخص مانند int

Pointer to function :

وقتی شما می خواهید به توابع به چشم data نگاه کنید و آن را تغییر بدید

نحوه اعلان تابع:

Void f(int)

نحوه اعلان پونتر به تابع :

Void (\*pf)(int)

Void (\*func\_ptr\_type) (int)

این خط یک پوینتر به تابع void با پارامتر int میسازد اما خط زیر func\_ptr\_type را به عنوان نوع پوینتر به این توابع تعریف می کند :

Typedef void (\*func\_ptr\_type) (int)