## Subject: By dropping MSC71 support, we can now activate generic pointers templates
Posted by mirek on Sat, 12 May 2012 13:05:09 GMT
View Forum Message <> Reply to Message

There are two of them:

```
template<class T> unsigned GetHashValue(T *)
```

and

```
template<class T> String AsString(T *)
```

means it is now not required to provide GetHashValue for pointers and pointers can be e.g. DUMPed directly.

## Subject: Re: By dropping MSC71 support, we can now activate generic pointers templates
Posted by kohait00 on Tue, 22 May 2012 16:34:53 GMT
View Forum Message <> Reply to Message

Hi mirek

i ran into a dumb problem since that..

this used to work (Python export of a class, say 'Size')

```
  .def("__str__", &::AsString<Size>)
```

now, TDMGCC451 still compiles it, MSC9 doesnt.
I could wrap this into an explicit differently named function, but this is a general problem.. how to get a pointer to that function?
casting to an explicit version doesnt work as it seems

```
  .def("__str__", (String (*)(const Size&))&AsString<Size>)
```

any hint?

## Subject: Re: By dropping MSC71 support, we can now activate generic pointers templates
Posted by mirek on Wed, 23 May 2012 06:14:00 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 22 May 2012 12:34Hi mirek i ran into a dumb problem since that..

this used to work (Python export of a class, say 'Size')

```
  .def("__str__", &::AsString<Size>)
```

now, TDMGCC451 still compiles it, MSC9 doesnt.
I could wrap this into an explicit differently named function, but this is a general problem.. how to get a pointer to that function?
casting to an explicit version doesnt work as it seems

```
  .def("__str__", (String (*)(const Size&))&AsString<Size>)
```

any hint?

This is really sad. IMO it is not 'our' bug....

What about adding AsStringConcrete template (same definition as AsString), which is NOT defined for pointers?

Mirek