Subject: [REJECTED]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by Oblivion on Sun, 21 Feb 2016 13:46:13 GMT
View Forum Message <> Reply to Message

Hello guys,

While SSH package is on its way, I would like to share some part of its code which might be in general a useful addition to U++.

VarArgs class is a special type of container, eliminating the need for C style variadic functions and related va_xxx macros. It can contain any type of argument in an orderly fashion, and allows dynamic manipulation of the arguments list.

This class was actually an integral part of my AsyncQueue class (a simple yet extremely flexible, generic synchronization tool for writing single-threaded, non-blocking apps/components in a uniform way, inspired by existing solutions in U++ framework) which I use building SSH package and will give you more information in the following days.

Below is the entire interface of the class:

```
class VarArgs : Moveable<VarArgs> {
   Vector<Any> args;
public:
   template<class T> VarArgs&  Add(const T& t)              { args.Add().Create<T>() = t; return
*this; }
   template<class T> VarArgs&  operator<<(const T& t)       { return Add(t); }
   template<class T> T&        Set(int i, const T& t)       { T& arg = args[i]; arg = t; return arg; }
   template<class T> T&        Insert(int i, const T& t)    { return args.Insert(i, t); }
   template<class T> T&        Get(int i)                   { ASSERT(Is<T>(i)); return args[i].Get<T>(); }
   inline void               Remove(int i)                { args.Remove(i); }
   inline int                GetCount() const             { return args.GetCount(); }
   inline void               Clear()                      { args.Clear(); }
   template<class T> bool      Is(int i) const            { args[i].Is<T>(); }
   inline bool               IsEmpty() const              { return args.IsEmpty(); }
   inline bool               IsPicked() const             { return args.IsPicked(); }

   VarArgs()                 {}
   virtual ~VarArgs()        {}
};
```

I added a very simple example to the package to show what this class is capable of:

```
#include <Core/Core.h>
#include <VarArgs/VarArgs.h>

using namespace Upp;

String str_cpp = "This is a demonstration of the VarArgs class of C++!";

bool PrintNumberViaGate(int i)
{
   Cout() << "Number printed via gate: " << i << "\r\n";
   return false;
}
void Foo(VarArgs& va)
{
   Cout() << va.Get<int>(0) << "\r\n";
   Cout() << va.Get<float>(1) << "\r\n";
   Cout() << va.Get<String>(2) << "\r\n";

   String* str_upp = va.Get<String*>(3);
   str_upp->Set(str_upp->ReverseFind('C'), 'U');
   Cout() << str_cpp << "\r\n";

   Gate1<int> cb = va.Get<Gate1<int> >(4);
   cb(9999);
}

CONSOLE_APP_MAIN
{
   Foo(VarArgs() << 10 << 22.33f << String("Hello World!") << &str_cpp <<
callback(PrintNumberViaGate));
//  Same thing can be done via explicit calls.
//  VarArgs va;
//  va  .Add(10)
//      .Add(22.33f)
//      .Add(String("Hello World!"))
//      .Add(&str_cpp)
//      .Add(callback(PrintNumberViaGate));
//  Foo(va);

}
```

Mirek, in case you find it useful, please feel free to modify and/or add it to the Core package. If you you reject it, I'd like to upload it to the Bazaar section.

Any suggestions, bug reports and criticism are welcome.

(Tested under Linux (ArchLinux) with GCC 5.3.4)

Regards,
Oblivion

File Attachments

1) VarArgs Package.zip, downloaded 334 times