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 371 times

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)

Posted by Novo on Sun, 21 Feb 2016 15:11:36 GMT

View Forum Message <> Reply to Message

Hi Oblivion,

You've added virtual destructor, which is IMHO not necessary. It will just add an extra pointer to your data structure.
I'm just curious, why you cannot use plain Vector<Any>?

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)

Posted by Oblivion on Sun, 21 Feb 2016 16:23:44 GMT

View Forum Message <> Reply to Message

Novo wrote on Sun, 21 February 2016 17:11Hi Oblivion,

You've added virtual destructor, which is IMHO not necessary. It will just add an extra pointer to your data structure.
I'm just curious, why you cannot use plain Vector<Any>?

Hello Novo,

Thank you for your comment.

Yes you are right, it is unnecessary in this case; a relic from its first version I've forgotten to delete.
Thanks for pointing out! :)

As to your question, well, there are mainly two reasons why I do not want to use plain Vector<Any>:

First, as you can see, VarArgs is a convenience class/wrapper. It simply hides the unnecessary interface/clutter of Vector, which is huge, and allows a rather goal oriented, limited interface.
This brings me to the second reason: semantic considerations. Think of VarArgs as something like the Id class, which is actually a String, in the Core package. But unlike Id class,
VarArgs has a specific set of methods, and announces its purpose: It stores arguments for methods, or functions in a dynamic manner. Nothing less, nothing more.

From a newcomers perspective, when all you want to do is to use variadic functions, whose macros are not very safe and trying to find an equivalent for them in U++, where would you look at first?
Would you read through the whole documentation, and try to figure out which combinations will work? (This is what ought to be, but we are living in a world where time is precious.) It would be time consuming to read through the whole documentation of Vector, which is rather huge, and Any. Having specific classes under your reach for widely used programming techniques is, I believe, the goal of rapid application development frameworks, and U++ usually excels at this.
Aside from that, it improves the source code readablity.

Regards,
Oblivion

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by Novo on Sun, 21 Feb 2016 18:42:36 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sun, 21 February 2016 11:23read through the whole documentation of Vector, which is rather huge, and Any
1) From the documentation: "Any is a special type of container capable of containing none or single element of any type.". Why do you want to pass no arguments with Any? I believe you wanted to use Value instead.
2) va_ macros do not allocate anything on heap. Vector does, and Value can do that. So, it can be more expensive to prepare arguments for a call, then just call a function, and there is also memory fragmentation ...
3) Vector has a set of methods which is standard for vector-like classes. Design of associative containers in Upp is quite unique though.

Just my two cents.

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by Oblivion on Sun, 21 Feb 2016 20:34:14 GMT
View Forum Message <> Reply to Message

Quote:1) From the documentation: "Any is a special type of container capable of containing none

or single element of any type.". Why do you want to pass no arguments with Any? I believe you wanted to use Value instead.
2) va_ macros do not allocate anything on heap. Vector does, and Value can do that. So, it can be more expensive to prepare arguments for a call, then just call a function, and there is also memory fragmentation ...
3) Vector has a set of methods which is standard for vector-like classes. Design of associative containers in Upp is quite unique though.

Hello Novo, thank you for taking time to criticising my proposal.

1) I don't want to pass no arguments. AFAIK, it depends on the method used. Here, it should be guaranteed to be created with content. Besides, I considered writing this class using Value, and as a matter of fact I wrote it. But soon I ran into problems. Value only returns a constant reference, where I also want to allow passing of non-const references, pointers including callbacks, that can be manipulated (although I can use const_cast. In fact, this give me an idea...).

2) A fair point. But I don't think the memory overhead will be unacceptable (I admit I have to run tests, though). There are use cases, especially in async programming where calling variadic functions using traditional ways can get too tricky. (after all, VarArgs was born of this reason).

3) I don't quite understand this one. While VarArgs is technically a container (I defined it as such in the description, I know), it isn't per se. It is supposed to be a convenince class with vector-like interface.

Regards,

Oblivion

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by mirek on Mon, 22 Feb 2016 19:34:23 GMT
View Forum Message <> Reply to Message

IMO Value makes better alternative if 'args' are concretes.

If not, C++11 parameter packs would probably be better. Or sometimes tuples.

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by Novo on Thu, 25 Feb 2016 04:10:35 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sun, 21 February 2016 15:34
3) I don't quite understand this one. While VarArgs is technically a container (I defined it as such

in the description, I know), it isn't per se. It is supposed to be a convenince class with vector-like interface.

I wanted to say that interface of Vector is very simple and standard. There is nothing unusual and hard to understand about it. Design of Index and ArrayIndex is somewhat unusual.
IMHO, nobody should have problems using plain Vector.

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by Oblivion on Thu, 25 Feb 2016 20:35:43 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 22 February 2016 21:34
IMO Value makes better alternative if 'args' are concretes.

If not, C++11 parameter packs would probably be better. Or sometimes tuples.

Hello Mirek,

Yes it certainly would be better in that case. But main focus of this class is on mutable pointers, and references. As I mentioned above, I actually decoupled it from the simple synchronization tool I wrote, where async jobs (methods with variable arguments) are queued to be executed later. It works well there.

Anyway, I am going to mark this proposal as rejected, for it now seems to me a bad idea to generalize this type of argument passing.

Quote:Oblivion wrote on Sun, 21 February 2016 15:34

3) I don't quite understand this one. While VarArgs is technically a container (I defined it as such in the description, I know), it isn't per se. It is supposed to be a convenince class with vector-like interface.

I wanted to say that interface of Vector is very simple and standard. There is nothing unusual and hard to understand about it. Design of Index and ArrayIndex is somewhat unusual.
IMHO, nobody should have problems using plain Vector.

Hello Novo,

Ah, I see. I got your point now. Thanks!

Regards,

Oblivion.

Subject: Re: [PROPOSAL]: VarArgs class for U++ (va_ macros replacement, in U++ stlye)
Posted by mirek on Fri, 26 Feb 2016 10:10:32 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 25 February 2016 21:35mirek wrote on Mon, 22 February 2016 21:34
IMO Value makes better alternative if 'args' are concretes.

If not, C++11 parameter packs would probably be better. Or sometimes tuples.

Hello Mirek,

Yes it certainly would be better in that case. But main focus of this class is on mutable pointers, and references. As I mentioned above, I actually decoupled it from the simple synchronization tool I wrote, where async jobs (methods with variable arguments) are queued to be executed later. It works well there.

Anyway, I am going to mark this proposal as rejected, for it now seems to me a bad idea to generalize this type of argument passing.

Ah, sorry, I might have been confused by the example, which is only using concretes.

I will reinvestigate with that in mind...

Mirek