Subject: ESC_STRING and other questions
Posted by Shire on Sat, 15 Mar 2008 09:37:12 GMT
View Forum Message <> Reply to Message

Is reasonable to add ESC_STRING (internally WString) type? Now strings are represented as arrays of ESC_NUMBER values, with large overhead and collisions when typecasting to StdValue.
Separate string type can faster execution. Strings can declare all array operations and explicit convertions to/from arrays.
Also, I think, that ESC_MAP's keys can be strictly ESC_STRING or ESC_NUMBER - other types have questionable usability. Key type is possible to detect at definition time.
I also think, that file operations in Standarrd library are unsafe. Would you separate them?
And the last question, will U++ team continue development of Esc?Esc language is very good to understandinh and embedding in application. Additionally, it fully open and compatible with UPP.

Subject: Re: ESC_STRING and other questions
Posted by mirek on Sat, 15 Mar 2008 10:13:23 GMT
View Forum Message <> Reply to Message

Shire wrote on Sat, 15 March 2008 05:37Is reasonable to add ESC_STRING (internally WString) type? Now strings are represented as arrays of ESC_NUMBER values, with large overhead and collisions when typecasting to StdValue.
Separate string type can faster execution. Strings can declare all array operations and explicit convertions to/from arrays.

That is true. However, this would break "typeless" design paradigm.

Anyway, the idea was to optimize EscValue so that it will store string data as (surprise) WString as long as possible. In fact, maybe even as String - that would only be better

However, so far, despite being ineffective, Esc implementation was "good enough" for target applications...

Quote:
Also, I think, that ESC_MAP's keys can be strictly ESC_STRING or ESC_NUMBER - other types have questionable usability. Key type is possible to detect at definition time.

Ditto It should be possible to optimize EscValue for this.

Quote:
I also think, that file operations in Standarrd library are unsafe. Would you separate them?

Unsafe? Why?

Quote:
And the last question, will U++ team continue development of Esc?Esc language is very good to understandinh and embedding in application. Additionally, it fully open and compatible with UPP.

Esc is an important part of several comercial applications, so you should not be afraid this is a dead end. We have to maintain it...

The only problem seems to be that generally, current implementation satisfies our need Means there is only a little reason to optimize Esc.

Mirek

Subject: Re: ESC_STRING and other questions
Posted by Shire on Sat, 15 Mar 2008 11:33:55 GMT
View Forum Message <> Reply to Message

Full typelessness is not so good. And absence of such fundamental type as string makes some things more different, for example, work with localization or path processing.
Well, I prefer WString - localized identificators will be good for applied programmers (such as in MS Excel).

IMHO, file (and other system) operations must not be accessible _by_default_ in scripts. Otherwise there are _default_ security hole - makes possible to stole or owerwrite files in such simple applications as expression calculator.

Well, your satisfaction is quite clear But IMHO, "polishing" Esc will be better than plugging in third-party scripting, such as squirrel or lua.

Subject: Re: ESC_STRING and other questions
Posted by mirek on Sat, 15 Mar 2008 12:51:49 GMT
View Forum Message <> Reply to Message

Shire wrote on Sat, 15 March 2008 07:33Full typelessness is not so good. And absence of such fundamental type as string makes some things more different, for example, work with localization or path processing.

Well, I guess we have to be more specific here.

What Esc does is to have only 3 basic value types - number, array of anything and map of anything. Ignoring effectivity, this covers everything you ever need...

I do not quite see how path processing is inferior to e.g. C. In fact, C seems to implement all strings as array of numbers just like Esc...

Quote:
IMHO, file (and other system) operations must not be accessible _by_default_ in scripts. Otherwise there are _default_ security hole - makes possible to stole or owerwrite files in such simple applications as expression calculator.

OK, good point. I guess we should divide to CoreStdLib (without file ops) and StdLib - correct?

Quote:
Well, your satisfaction is quite clear But IMHO, "polishing" Esc will be better than plugging in third-party scripting, such as squirrel or lua.

I agree. Well, if you have time, you can try to optimize EscValue

Mirek

Subject: Re: ESC_STRING and other questions
Posted by Shire on Sat, 15 Mar 2008 17:53:34 GMT
View Forum Message <> Reply to Message

Quote:
What Esc does is to have only 3 basic value types - number, array of anything and map of anything.
Ignoring effectivity, this covers everything you ever need...

User cannot differ array of numbers and string, and cannot debug wrong conversion. Strings can have stronger restrictions and easy native processing routines (without double conversation to and from WString)

Quote:
I do not quite see how path processing is inferior to e.g. C. In fact, C seems to implement all strings as array of numbers just like Esc...

(in C++) this pointer incapsulated in typed std::string or String. Compare String and Vector<char>

Quote:
OK, good point. I guess we should divide to CoreStdLib (without file ops) and StdLib - correct?

Well, I prefer more modular structure. Like Std, StreamIO, FileIO, Codecs, DateTime, SQL, etc. It can help apply restrictions easier.

Quote:

Well, if you have time, you can try to optimize EscValue

Yes, I'll try it.

Subject: Re: ESC_STRING and other questions
Posted by mirek on Sat, 15 Mar 2008 20:37:46 GMT
View Forum Message <> Reply to Message

Shire wrote on Sat, 15 March 2008 13:53Quote:
What Esc does is to have only 3 basic value types - number, array of anything and map of anything.
Ignoring effectivity, this covers everything you ever need...

User cannot differ array of numbers and string, and cannot debug wrong conversion. Strings can have stronger restrictions and easy native processing routines (without double conversation to and from WString)

Well, that is true, but we are not creating another C++, are we?

This is supposed to be a simple language for simple scripts. If you are running into type issues, you have choosen the wrong
tool..

Quote:
Well, I prefer more modular structure. Like Std, StreamIO, FileIO, Codecs, DateTime, SQL, etc. It can help apply restrictions easier.

Come on. "Esc" standard library has only about 20 functions total...

Sure, you can easily add more, but to isolate "dangerous" File I/O, what I suggest is enough IMO.

Mirek

Subject: Re: ESC_STRING and other questions
Posted by Shire on Sun, 16 Mar 2008 09:18:41 GMT
View Forum Message <> Reply to Message

Quote:
Well, that is true, but we are not creating another C++, are we?

This is supposed to be a simple language for simple scripts. If you are running into type issues,

you have choosen the wrong
tool..

Well, simple script languages must not force user to worry about types. But even non-professional applied programmers differ string and array. Strings have many specific operations, like toupper, find-and-replace, regexp, etc, which is useless on arrays. Additionally, IMHO, majority of simple scripts work with strings hard, and I cannot imagine script language without native string type with standard operations.

Quote:
Sure, you can easily add more, but to isolate "dangerous" File I/O, what I suggest is enough IMO.

Yes, it will be enough.

Optimization of map can be done in three ways:

1. Introduce new internal type, ESC_DICTionary - internally VectorMap<String, EscValue> - the most effective way, it can remove the difference between global and map.
2. Initialize map with EscDictionary (derived from EscMap and convert it to EscMap when user appends first non-string or non-number key. It is difficult to determine string and non-string.
3. Introduce new type ESC_STRING. Can be combined with 2.

IMHO 3. is the best.

Subject: Re: ESC_STRING and other questions
Posted by mirek on Sun, 16 Mar 2008 09:41:51 GMT
View Forum Message <> Reply to Message

Shire wrote on Sun, 16 March 2008 05:18Quote:
Well, that is true, but we are not creating another C++, are we?

This is supposed to be a simple language for simple scripts. If you are running into type issues, you have choosen the wrong
tool..

Well, simple script languages must not force user to worry about types. But even non-professional applied programmers differ string and array. Strings have many specific operations, like toupper, find-and-replace, regexp, etc, which is useless on arrays.

Actually, I agree with toupper, but IMO find-and-replace or even regexp are not that useless for arrays...

Quote:
Additionally, IMHO, majority of simple scripts work with strings hard, and I cannot imagine script language without native string type with standard operations.

You do not need to imagine. Esc is one

Quote:
Optimization of map can be done in three ways:

1. Introduce new internal type, ESC_DICTionary - internally VectorMap<String, EscValue> - the most effective way, it can remove the difference between global and map.
2. Initialize map with EscDictionary (derived from EscMap and convert it to EscMap when user appends first non-string or non-number key. It is difficult to determine string and non-string.
3. Introduce new type ESC_STRING. Can be combined with 2.

Obviously That was the plan:

Keep array data in String as long as possible, then move to WString, then perhaps Vector<int> or even Vector<double> or Vector<String> or Vector<WString>....

Keep map data in VectorMap<String, EscValue> or VectorMap<int, EscValue> as long as possible.

But do not waste more than 1000 lines of C++ on this

Would be also nice to have some benchmark to prove the approach is really viable.

Mirek

Subject: Re: ESC_STRING and other questions
Posted by galious on Fri, 16 May 2008 11:44:26 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 16 March 2008 10:41Shire wrote on Sun, 16 March 2008 05:18

Well, simple script languages must not force user to worry about types. But even non-professional applied programmers differ string and array. Strings have many specific operations, like toupper, find-and-replace, regexp, etc, which is useless on arrays.

Actually, I agree with toupper, but IMO find-and-replace or even regexp are not that useless for arrays...

I don't even agree with toupper. toupper can be viewed as tr(anslate|ansliterate) which can be found in Perl or sed and many other languages. tr on it's turn can be viewed as a specialised version of find-and-replace, which on its turn does have its use for arrays.

Quote:Quote:
Additionally, IMHO, majority of simple scripts work with strings hard, and I cannot imagine script language without native string type with standard operations.

You do not need to imagine. Esc is one

Euphoria with it's concept of sequences and atoms is another one. I'm probably one of only a few people who like as less as possible build-in types of which one can create it's own types.

Martin

Subject: Re: ESC_STRING and other questions
Posted by mirek on Fri, 23 May 2008 07:46:45 GMT
View Forum Message <> Reply to Message

galious wrote on Fri, 16 May 2008 07:44
Euphoria with it's concept of sequences and atoms is another one. I'm probably one of only a few people who like as less as possible build-in types of which one can create it's own types.
Martin

Actually, Euphoria was sort of inspiration for Esc... (among many other influences).

Mirek