Subject: Format 1234567.89 as 1,234,567.89
Posted by Lance on Wed, 13 Feb 2013 20:17:26 GMT
View Forum Message <> Reply to Message

In examples, there is a package called Format which demonstrates some typical use of Format, where:

Format("%nl",1234567.89);

would generate the desired format.

However, the above produces a very different result when GUI is defined. Is it because of a bug in Core, or there is a more reliable and consistant format string for that purpose?

Thanks,

Lance

Subject: Re: Format 1234567.89 as 1,234,567.89
Posted by Lance on Sat, 16 Feb 2013 19:46:13 GMT
View Forum Message <> Reply to Message

I figured it out.

For some reason, the language (result of GetLangName(GetCurrentLanguage())) is initialized to English for UPP console application, the same is not done for GUI application.

Subject: Re: Format 1234567.89 as 1,234,567.89
Posted by Lance on Sat, 30 Nov 2013 15:30:28 GMT
View Forum Message <> Reply to Message

BTW, if you want to format a double to certain decimal digits plus thousand separator, aside from the language setting step mentioned above, the format string should be something like this:

"%2!nl"
where 2 is the number of decimal digits, ! will force trailing 0 padding if needed, nl will allow localized thousand separator.

Happy coding!

Subject: Re: Format 1234567.89 as 1,234,567.89
Posted by Lance on Mon, 09 Dec 2013 00:52:15 GMT
View Forum Message <> Reply to Message

As of today, the Format("%nl",1234.56) part doesn't work on Ubuntu . It will output 123456. I assume it's because neither decimal point character and thousand separator character has been properly initialized. And yes, I am talking about the output of the Format program shipped with each Upp release in which SetLanguage has been called.

Subject: Re: Format 1234567.89 as 1,234,567.89
Posted by Lance on Mon, 09 Dec 2013 03:49:29 GMT
View Forum Message <> Reply to Message

Which leads us to this question: how to write a customized formatter? The designers of U++ have actually provided quite wisely-designed way to expand the Format() functionality, even though it's not very well documented or demonstrated. You need to read the relevant library codes to understand it.

Instead of tolerating inconsistent/incorrect behavior and wait for somebody to fix the library, we can create our own customized Formatter and register it with the Core Formatting system.

Let's say, I want to create a MoneyFormatter to format numbers to certain decimal points and with thousand separator. And I have chosen 'm' as the format id. To keep it simple and stupid, we will allow specifying number of decimal points only. Here is some example of using the 'm' format

%m, format to 2 decimal points (default value);
%5m, format to 5 decimal points (padding with trailing 0s if needed)

Here is the required code, which can all be in the same cpp file, no declaration in any header file is needed.

```
static void sRegisterFormatters()
{
 ONCELOCK {
  RegisterNumberFormatter("m",  MoneyFormatter);
 }
}

INITBLOCK {
 sRegisterFormatters();
}
String Format(double number, unsigned decimal, const char * zeroAs=NULL);
```

```
String MoneyFormatter(const Formatting& f)
{
 if(IsNull(f.arg))
  return String();
 double value = f.arg;
 const char *s = f.format;
 int digits = 2;
 if(IsDigit(*s) || *s == '-' && IsDigit(s[1])) {
  digits = (int)strtol(s, NULL, 10);
  while(IsDigit(*++s))
   ;
 }
 return Format(value, digits);
}

String Format(double number, unsigned decimal, const char * zeroAs)
{
 unsigned long long ipart,fract,mult=1ULL;
 String s;
 char buff[20];
 bool sign=number<0;

 if(IsNull(number))
  number=0.;

 if(number<-1e10 || number>1e10)
  return String("ERROR");

 if(sign)
  number=-number;

 double f=.5;

 for(unsigned i=0; i<decimal; i++){
  mult*=10;
  f/=10.;
 }
 number+=f;
 ipart=(unsigned long long)number;
 fract=(unsigned long long)((number-ipart)*mult);
 if(ipart==0uLL && fract==0uLL && zeroAs!=NULL)
  return s<<zeroAs;

 if(decimal>0)
  s<<fract;

 if((unsigned)s.GetLength()<decimal)
```

```
  s=String('0', decimal-s.GetLength()).Cat()<<s;

 if(decimal>0)
  s='.'+s;
 if(ipart==0)
  s="0"+s;
 while(ipart)
 {
  int j=ipart%1000;
  ipart /=1000;
  if(ipart)
  {
   sprintf(buff,",%03d", j);
  }else{
   sprintf(buff, "%d", j);
  }
  s=buff+s;
 }
 if(sign)
  s="-"+s;
 return s;
}
```