
Subject: again... use if deleted function :?

Posted by [idkfa46](#) on Sun, 14 Jan 2018 11:10:53 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi guys,

I'm here asking for your help again because I have one more error compiling my code.

Error:

```
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibVerifiche\Tensoflessione.cpp (339):  
error: use of deleted function 'QTFStr::QTFStr(const QTFStr&)'  
(): return q tf ;  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): note:  
'QTFStr::QTFStr(const QTFStr&)' is implicitly deleted because the default definition would be  
ill-formed:  
(): class Q TFStr  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<QTFStr::QTFFormat>::Vector(const  
Upp::Vector<QTFStr::QTFFormat>&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<int>::Vector(const Upp::Vector<int>&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<int>::Vector(const Upp::Vector<int>&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<Upp::Vector<int> >::Vector(const  
Upp::Vector<Upp::Vector<int> >&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibVerifiche\Tensoflessione.cpp (423):  
error: use of deleted function 'QTFStr::QTFStr(const QTFStr&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibVerifiche\Flessione.cpp (409): error:  
use of deleted function 'QTFStr::QTFStr(const QTFStr&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<QTFStr::QTFFormat>::Vector(const  
Upp::Vector<QTFStr::QTFFormat>&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<int>::Vector(const Upp::Vector<int>&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<int>::Vector(const Upp::Vector<int>&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibQTF/QTFStr.h (13): error: use of  
deleted function 'constexpr Upp::Vector<Upp::Vector<int> >::Vector(const  
Upp::Vector<Upp::Vector<int> >&)'  
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibVerifiche\Flessione.cpp (508): error:  
use of deleted function 'QTFStr::QTFStr(const QTFStr&)'
```

function:

```
QTFStr Tensoflessione::Instabilitaqtf(QTFStr &qtf)  
{
```

```

String risultato;

<= 1 &";

<= 1 ";
risultato << GetVerificaInst();

qtf.StartTable(3,2,2,10).CharSize(12);
qtf.TableSubTitle("Instabilità flesso-torsionale");
qtf.LeftCellMargin(75)("I[, eff]").LeftCellMargin(15).AlignRight()
  (Format("%.2f", GetInstab().leff)).AlignLeft("mm")("Lunghezza libera di inflessione");

  (Format("%.2f", GetInstab().sigmam_critico)).AlignLeft("")("Tensione critica di svergolamento");

  (Format("%.2f", GetInstab().lambda_relm)).AlignLeft("")("Snellezza relativa di svergolamento");
qtf.LeftCellMargin(75)("k[, crit]").LeftCellMargin(15).AlignRight()
  (Format("%.2f", GetInstab().kcrit)).AlignLeft("N/mm[` 2]")("Coefficiente di sbandamento
laterale");
qtf.TableSubTitle(risultato);
qtf.EndTable();

qtf.TableSubTitle(risultato);
qtf.EndTable();

return qtf;
}

```

QTFStr class:

```

class QTFStr
{
public:
  class QTFFormat : public Moveable<QTFFormat>
  {
  public:
    int tableKeep;
    bool tableBorder;
    bool bold;
    bool strike;
    bool underline;
    bool subscript;
    bool superscript;
    bool italic;
    int align;
    int charSize;
    char font;
    int bottomCellMargin;
  };
};

```

```

int topCellMargin;
int leftCellMargin;
int rightCellMargin;
String foreColor;
String backColor;
String cellbackground;

QTFFormat();
void Clear(void);
QTFFormat(const QTFFormat &fmt);
QTFFormat const &operator=(QTFFormat const &fmt);
};

```

private:

```

// QTF string container
String str;

// settings stack
Vector<QTFFormat> formatStack;

// table depth level
int tableLevel;

// table running values
Vector<int>  tableColumns;
Vector<int>  tableColumn;
Vector<Vector<int> > columnWidths;;

// formatting running values
QTFFormat format;

// error flag -- stops any processing if set
bool err;

// outputs format codearound string
String FormatString(const String &s);

```

protected:

public:

```

// constructor
QTFStr();

// empty string and reset all formats to default
QTFStr &Clear(void);

```

```

// clears all formatting options -- reset them to defaults
QTFStr &ClearFormats(void);

// push/pop format values
QTFStr &PushFormat(void);
QTFStr &PopFormat(void);

// set/get format
QTFStr &GetFormat(QTFFormat &fmt) { fmt = format; return *this; }
QTFStr &SetFormat(QTFFormat const &fmt) { format = fmt; return *this; }

// checks condition, if true sets error value, logs the (first) error with its QTF
// and stops any further processing. Returns err flag
bool CheckError(bool cond, String const &msg);

// format change functions
QTFStr &TableOnPage(void) { format.tableKeep = KEEP_TABLE_ON_PAGE; return *this; }
QTFStr &CellOnPage(void) { format.tableKeep = KEEP_CELL_ON_PAGE; return *this; }
QTFStr &FreeTable(void) { format.tableKeep = FREE_TABLE; return *this; }
QTFStr &TableBorder(void) { format.tableBorder = true; return *this; }
QTFStr &NoTableBorder(void) { format.tableBorder = false; return *this; }
QTFStr &Bold(void) { format.bold = true; return *this; }
QTFStr &NoBold(void) { format.bold = false; return *this; }
QTFStr &Strike(void) { format.strike = true; return *this; }
QTFStr &NoStrike(void) { format.strike = false; return *this; }
QTFStr &Underline(void) { format.underline = true; return *this; }
QTFStr &NoUnderline(void) { format.underline = false; return *this; }
QTFStr &Subscript(void) { format.subscript = true; return *this; }
QTFStr &NoSubscript(void) { format.subscript = false; return *this; }
QTFStr &SuperScript(void) { format.superscript = true; return *this; }
QTFStr &NoSuperScript(void) { format.superscript = false; return *this; }
QTFStr &Italic(void) { format.italic = true; return *this; }
QTFStr &NoItalic(void) { format.italic = false; return *this; }
QTFStr &AlignLeft(void) { format.align = ALIGN_LEFT; return *this; }
QTFStr &AlignCenter(void) { format.align = ALIGN_CENTER; return *this; }
QTFStr &AlignRight(void) { format.align = ALIGN_RIGHT; return *this; }
QTFStr &AlignJustify(void) { format.align = ALIGN_JUSTIFY; return *this; }
QTFStr &CharSize(int siz) { format.charSize = siz; return *this; }
QTFStr &IncCharSize() { if (format.charSize < 9) format.charSize++; return *this; }
QTFStr &DecCharSize() { if (format.charSize > 0) format.charSize--; return *this; }
QTFStr &Font(char fnt) { format.font = fnt; return *this; }
QTFStr &ForeColor(String col = "") { format.foreColor = (col == "" ? "0" : col); return *this; }
QTFStr &BackColor(String col = "") { format.backColor = (col == "" ? "2" : col); return *this; }
//
QTFStr &BackgroundColor(String col = ""){ format.cellbackground = (col == "" ? "2" : col); return
*this; }
//
QTFStr &LeftCellMargin(int m = 25) { format.leftCellMargin = m; return *this; }

```

```

QTFStr &RightCellMargin(int m = 25) { format.topCellMargin = m; return *this; }
QTFStr &TopCellMargin(int m = 15) { format.rightCellMargin = m; return *this; }
QTFStr &BottomCellMargin(int m = 15) { format.bottomCellMargin = m; return *this; }

```

```

// output text
QTFStr &Txt(const String &s, int merge = 0);
QTFStr &Txt(RichObject const &obj, int merge = 0);
QTFStr &operator()(const String &s, int merge = 0) { return Txt(s, merge); }
QTFStr &operator()(RichObject const &obj, int merge = 0) { return Txt(obj, merge); }

```

```

QTFStr &Iml(String const &Name, int cx, int cy, bool keepAspect = true, int merge = 0);
QTFStr &CenterIml(String const &Name, int cx, int cy, bool keepAspect = true, int merge = 0);

```

```

QTFStr &Object(RichObject const &obj, int merge = 0);
QTFStr &CenterObject(RichObject const &obj, int merge = 0);

```

```

// page title
QTFStr &PageTitle(String const &s1, String const &s2);

```

```

//////////////////////////////////////
//                                TABLE HANDLING                                //
//////////////////////////////////////
QTFStr &StartTable(Vector<int> const &colW);
QTFStr &StartTable(
    int w1 = Null, int w2 = Null, int w3 = Null, int w4 = Null,
    int w5 = Null, int w6 = Null, int w7 = Null, int w8 = Null);
QTFStr &TableRestart(void);
QTFStr &Cell(String const &Txt, int merge = 0);
QTFStr &TableTitle(String const &title);
QTFStr &TableSubTitle(String const &title);
QTFStr &TableNewLine(void);
QTFStr &EndTable(void);

operator const char *() { return str; }
operator const String&() { return str; }
String const &ToString(void) const { return str; }

}; // END Class QTFStr

```

The move constructor is implicitly deleted because the copy-constructor is explicitly deleted?

How can I easy fix it?

Regards,
Matteo

Subject: Re: again... use if deleted function :?
Posted by [idkfa46](#) on Sun, 14 Jan 2018 11:22:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

```
// constructor  
QTFStr() = default;
```

it shouldn't be a solution?

Matteo

Subject: Re: again... use if deleted function :?
Posted by [Oblivion](#) on Thu, 18 Jan 2018 10:26:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quote:
// constructor
QTFStr() = default;

it shouldn't be a solution?

Matteo

Hello Matteo,

There is a problem here:

```
QTFStr Tensoflessione::Instabilitaqtqf(QTFStr &qtf)  
{  
    // -----  
    //   ^  
    // You are passing a reference (which can be modified, as you do below.
```

String risultato;

```
<= 1 &"
```

```
<= 1 "  
risultato << GetVerificaInst();
```

```
qtf.StartTable(3,2,2,10).CharSize(12);  
qtf.TableSubTitle("Instabilità flesso-torsionale");  
qtf.LeftCellMargin(75)("I[, eff]").LeftCellMargin(15).AlignRight()
```

```

(Format("%.2f", GetInstab().leff)).AlignLeft("mm")("Lunghezza libera di inflessione");

(Format("%.2f", GetInstab().sigmam_critico)).AlignLeft("")("Tensione critica di svergolamento");

(Format("%.2f", GetInstab().lambda_relm)).AlignLeft("")("Snellezza relativa di svergolamento");
qtf.LeftCellMargin(75)("k[, crit]").LeftCellMargin(15).AlignRight()
(Format("%.2f", GetInstab().kcrit)).AlignLeft("N/mm[2"]("Coefficiente di sbandamento
laterale");
qtf.TableSubTitle(risultato);
qtf.EndTable();

qtf.TableSubTitle(risultato);
qtf.EndTable();

return qtf;
//-----
// ^
// Then, compiler attempts (by default) to copy qtf (But you need to explicitly define a copy
constructor, which is missing, and in it, explicitly copy the members of your class (using clone() ),
They are causing the error.)
// By the way, returning a QTFStr from this function shoudn't be necessasy at all, since you
are already modifying the referenced instance! (it is non-const), unless you really need a copy of
it. It is a much cheaper solution.
}

```

OR, if you don't use the referenced variable qtf after calling the method but you need your method to return a QTFStr, you can move it instead.

```

[
    QTFStr(QTFStr&&) = default;
    QTFStr& operator=(QTFStr&&) = default;

```

Then,

```

return pick(qtf);

```

And please read about copy and move semantics of both C++11 (StackOverFlow has plenty of good articles on it), and UPP.

Best regards,
Oblivion

Subject: Re: again... use if deleted function :?
Posted by [idkfa46](#) on Sun, 21 Jan 2018 11:43:50 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thank you Oblivion,
I read something about copy and move semantics and I solved my issue defining a copy constructor as you suggested.

By the way is not clear for me when you write:

Quote:

// By the way, returning a QTFFStr from this function shouldn't be necessary at all, since you are already modifying the referenced instance! (it is non-const), unless you really need a copy of it. It is a much cheaper solution.

What is the much cheaper solution?

And then, how can I define a copy constructor for a Template<class T> function such as:

```
template<class T> void SortDurata(VectorMap<int, T> &Load)
{
    //Riordina VectorMap in ordine crescente
    Vector<int> keys = Load.PickKeys();
    Vector<T> values = Load.GetValues();
    IndexSort(keys, values);
    Load.Clear();
    Load = VectorMap<int, T>(keys, values);
}
```

```
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\TabellaCarichi.cpp (192):
error: use of deleted function 'Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >&
Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >::operator=(const Upp::VectorMa
p<Upp::String, Upp::Vector<Upp::Value> >&)'
C:\upp\uppsrc\Core\Map.h (193): error: use of deleted function 'Upp::AMap<Upp::String,
Upp::Vector<Upp::Value>, Upp::Vector<Upp::Vector<Upp::Value> > >& Upp::AMap<Upp::String,
Upp::Vector<Upp::Value>, U
pp::Vector<Upp::Vector<Upp::Value> > >::operator=(const Upp::AMap<Upp::String,
Upp::Vector<Upp::Value>, Upp::Vector<Upp::Vector<Upp::Value> > >&)'
C:\upp\uppsrc\Core\Map.h (33): error: use of deleted function 'Upp::Index<Upp::String>&
Upp::Index<Upp::String>::operator=(const Upp::Index<Upp::String>&)'
C:\upp\uppsrc\Core\Index.h (80): note: 'Upp::Index<Upp::String>&
Upp::Index<Upp::String>::operator=(const Upp::Index<Upp::String>&)' is implicitly declared as
deleted because 'Upp::Index<Upp::String>' declares a
move constructor or move assignment operator
(): class Index : MoveableAndDeepCopyOption<Index<T>> {
C:\upp\uppsrc\Core\Map.h (33): error: use of deleted function 'constexpr
```



```
Upp::Vector<Upp::Vector<Upp::Value> >& Upp::Vector<Upp::Vector<Upp::Value>
>::operator=(const Upp::Vector<Upp::Vector<Upp::Value>
>&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\TabellaCarichi.cpp (195):
error: use of deleted function 'Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >&
Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >::operator=(const Upp::VectorMa
p<Upp::String, Upp::Vector<Upp::Value> >&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (112): error:
use of deleted function 'constexpr Upp::Vector<_G>::Vector(const Upp::Vector<_G>&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (115): error:
cannot bind rvalue reference of type 'Upp::Vector<int>&&' to lvalue of type 'Upp::Vector<int>'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (112): error:
use of deleted function 'constexpr Upp::Vector<_Q>::Vector(const Upp::Vector<_Q>&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (115): error:
cannot bind rvalue reference of type 'Upp::Vector<int>&&' to lvalue of type 'Upp::Vector<int>'
```

Best regards,
Matteo

Subject: Re: again... use if deleted function :?
Posted by [Klugier](#) on Fri, 26 Jan 2018 21:08:21 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello,

How do you want to define copy constructor for the function? It is impossible, you can only define copy constructor for template type T and try to recompile your program. You should also provide information what is T in template generation.

We need more information which line is that:

TabellaCarichi.cpp (192):

Sincerely,
Klugier

Subject: Re: again... use if deleted function :?
Posted by [idkfa46](#) on Sat, 27 Jan 2018 13:25:46 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Klugier,
thank you for continue supporting me.

here below more details:

TabellaCarichi.cpp (rows 192 and 195):

```
void TabellaCarichi::AcceptedGrid()
{
    SLU.Clear();
    SLE.Clear();
    qtf_loads.Clear();

    if(!slu)
    {
        TabellaCarichiQtf();

        // Combinatore di carichi
        Combiner combiner;
        combiner.LoadGrid(Soll_CaratGrid);
        // Slu
        combiner.CombineSLU();
        SLU = combiner.GetSluVector();      <<<< ROW 192
        // Sle
        combiner.CombineSLE();
        SLE = combiner.GetSleVector();      <<<< ROW 195
    }
    else
    {
        for(int r=0; r< Soll_SluGrid.GetCount(); r++)
            SLU.Add(Soll_SluGrid.Get(r, TAG_DurataCARICO)) << "" << "" << Soll_SluGrid.Get(r,
TAG_SOLLECITAZIONI);

        // Scrivi tabella dei carichi SLU
        TabellaCarichiQtf();
    }
}
```

combiner.cpp:

```
void Combiner::LoadGrid(GridCtrl &grid)
{
    G1.Clear(); G2.Clear(); Q.Clear();

    for(int i=0; i < grid.GetCount(); i++)
    {
```

```

String cdc    = grid.Get(i, TAG_CondCARICO);
String durata = grid.Get(i, TAG_DurataCARICO);
String cat    = grid.Get(i, TAG_CatCARICO);
double sollecitaz = grid.Get(i, TAG_SOLLECITAZIONI);
int key       = DurConvToInt(durata);

if(cat == TAG_G1)
{
    G1.Add(key, _G(cdc, durata, cat, sollecitaz));
}
else if(cat == TAG_G2)
{
    G2.Add(key, _G(cdc, durata, cat, sollecitaz));
}
else
{
    Impostazioni &s = globalImpostazioni();
    double psi0j = s.GetPsi0j(cat);
    double psi1j = s.GetPsi1j(cat);
    double psi2j = s.GetPsi2j(cat);

    Q.Add (key, _Q(cdc, durata, cat, sollecitaz, psi0j, psi1j, psi2j));
}
}
::SortDurata(G1);      <<<< call to template<class T> (combiner.h)
::SortDurata(G2);      <<<< call to template<class T> (combiner.h)
::SortDurata(Q);       <<<< call to template<class T> (combiner.h)

}

```

combiner.h:

```

struct _G : public Moveable<_G> {
    String cdc;
    String durata;
    String categoria;
    double sollecitazione;

    _G(String s1, String s2, String s3, double d) : cdc(s1), durata(s2), categoria(s3),
sollecitazione(d) {}
};

struct _Q : public Moveable<_Q> {
    String cdc;
    String durata;
    String categoria;

```

```

double sollecitazione;
double Psi0j;
double Psi1j;
double Psi2j;

_Q(String s1, String s2, String s3, double d1, double d2, double d3, double d4): cdc(s1),
durata(s2), categoria(s3), sollecitazione(d1), Psi0j(d2), Psi1j(d3), Psi2j(d4) {}
};

struct _Output : public Moveable<_Output> {
    String durata;
    String txt;
    double tot;

    _Output(String s1, String s2, double d): durata(s1), txt(s2), tot(d) {}
};

class Combiner {
    // variabili - forse da eliminare
    String g1_txt, g2_txt, q_txt;
    double g1_tot, g2_tot, q_tot;

    // vettore dei carichi
    VectorMap<int, _G> G1;    // Carichi permanenti
    VectorMap<int, _G> G2;    // Carichi permanenti non strut
    VectorMap<int, _Q> Q;     // Carichi variabili

    // matrici carichi variabili
    VectorMap<int, Vector<double> > loads;
    VectorMap<int, Vector<double> > coefs;
    Vector<int> pos;

    // vettore dei risultati
    VectorMap<String, Vector<Value> > SLU;
    VectorMap<String, Vector<Value> > SLE;

    // calcolo combinazioni
    bool Combine(int i, int sl); // i=durata - sl-> 0=SLU, 1=SLErare, 2= SLEfreq, 3=SLEqperm
    void calcola_slu(int i); // i= durata del carico
    void calcola_sle_rara();
    void calcola_sle_freq();
    void calcola_sle_qperm();

    //
    int DurConvToInt(String s);
    String DurConvToStrin(int i);
    void PlusCdC(String &s){if(!s.IsEmpty()) s << "+";}

```

```

String StringaCdC(double gammaG1, double gammaG2, double gammaQ);
void clean(){ g1_txt.Clear(); g2_txt.Clear(); q_txt.Clear(); g1_tot = 0; g2_tot = 0; q_tot = 0; }

public:
// carica grid - tradizionale o multicolonna
void LoadGrid(GridCtrl &grid);
void LoadGrid(GridCtrl &grid, Id col);

// attiva combinazione
void CombineSLU(); // Stato limite ultimo
void CombineSLE(); // Stato limite di esercizio

// vettore dei risultati
VectorMap<String, Vector<Value> > &GetSluVector() { return SLU; }      <<<< pick(SLU) ???
VectorMap<String, Vector<Value> > &GetSleVector() { return SLE; }      <<<< pick(SLE) ???

double GetSluLoads(String durata);
String GetSluTxt(String durata);
double GetSleLoads(String combinazione);
String GetSleTxt(String combinazione);

// Tabelle Qtf
String qtf_SLU();
String qtf_SLE();
String qtf_SLErare();
String qtf_SLEfreq();
String qtf_SLEqperm();

typedef Combiner CLASSNAME;
Combiner(){}

Combiner(Combiner&&) = default;
Combiner& operator=(Combiner&&) = default;
};

template<class T> void SortDurata(VectorMap<int, T> &Load)
{
//Riordina VectorMap in ordine crescente - durata di carico
Vector<int> keys = Load.PickKeys();
Vector<T> values = Load.GetValues();
IndexSort(keys, values);
Load.Clear();
Load = VectorMap<int, T>(keys, values);
}

template<class T> void CheckDurata(bool &check, VectorMap<int, T> &X, int n, int i)
{
if(X.GetKey(n) == i)

```

```

    check = true;
}

```

Compiler error:

```

C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\TabellaCarichi.cpp (192):
error: use of deleted function 'Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >&
Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >::operator=(const Upp::VectorMa
p<Upp::String, Upp::Vector<Upp::Value> >&)'
    (): SLU = combiner.GetSluVector( ) ;
C:\upp\uppsrc\Core\Map.h (193): note: 'Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value>
>& Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >::operator=(const
Upp::VectorMap<Upp::String, Upp::Vec
tor<Upp::Value> >&)' is implicitly deleted because the default definition would be ill-formed:
    (): class V ectorMap : public MoveableAndDeepCopyOption<VectorMap<K, T>>,
C:\upp\uppsrc\Core\Map.h (193): error: use of deleted function 'Upp::AMap<Upp::String,
Upp::Vector<Upp::Value>, Upp::Vector<Upp::Vector<Upp::Value> > >& Upp::AMap<Upp::String,
Upp::Vector<Upp::Value>, U
pp::Vector<Upp::Vector<Upp::Value> > >::operator=(const Upp::AMap<Upp::String,
Upp::Vector<Upp::Value>, Upp::Vector<Upp::Vector<Upp::Value> > >&)'
C:\upp\uppsrc\Core\Map.h (33): error: use of deleted function 'Upp::Index<Upp::String>&
Upp::Index<Upp::String>::operator=(const Upp::Index<Upp::String>&)'
C:\upp\uppsrc\Core\Map.h (33): error: use of deleted function 'constexpr
Upp::Vector<Upp::Vector<Upp::Value> >& Upp::Vector<Upp::Vector<Upp::Value>
>::operator=(const Upp::Vector<Upp::Vector<Upp::Value>
>&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\TabellaCarichi.cpp (195):
error: use of deleted function 'Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >&
Upp::VectorMap<Upp::String, Upp::Vector<Upp::Value> >::operator=(const Upp::VectorMa
p<Upp::String, Upp::Vector<Upp::Value> >&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (113): error:
use of deleted function 'constexpr Upp::Vector<_G>::Vector(const Upp::Vector<_G>&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (116): error:
cannot bind rvalue reference of type 'Upp::Vector<int>&&' to lvalue of type 'Upp::Vector<int>'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (113): error:
use of deleted function 'constexpr Upp::Vector<_Q>::Vector(const Upp::Vector<_Q>&)'
C:\Users\Matteo\Dropbox\2_Sviluppo++\Workspace_upp\LibCombiner\Combiner.h (116): error:
cannot bind rvalue reference of type 'Upp::Vector<int>&&' to lvalue of type 'Upp::Vector<int>'

```

Best regards,
Matteo

Subject: Re: again... use if deleted function :?

Posted by [Klugier](#) on Sat, 27 Jan 2018 14:14:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello,

I believe you should clone or pick while assigning to SLU & SLE variables:

```
SLU = pick(combiner.GetSluVector());          <<<< ROW 192
// Sle
combiner.CombineSLE();
SLE = clone(combiner.GetSleVector());          <<<< ROW 195
```

The bad thing here is the code design:

```
VectorMap<String, Vector<Value> > &GetSluVector() { return SLU; }          <<<< pick(SLU) ???
VectorMap<String, Vector<Value> > &GetSleVector() { return SLE; }          <<<< pick(SLE) ???

// Add const VectorMap<String, Vector<Value>>& GetSluVector { return SLU; }
// Now somebody can call GetSluVector().Clear() that destroys your class encapsulation. The pick
operation would be disallow.

// In c++11 you do not need to make space after > in template, so ">>" is valid.

// Please also consider writing your code in plain English. It can be hard for the first time, but it will
percent in the future.
```

Sincerely,
Klugier

Subject: Re: again... use if deleted function :?

Posted by [idkfa46](#) on Sat, 27 Jan 2018 18:11:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thank you Klugier!

I applied the following changes too:

```
template<class T> void SortDurata(VectorMap<int, T> &Load)
{
    //Riordina VectorMap in ordine crescente - durata di carico
    Vector<int> keys = Load.PickKeys();
    Vector<T> values = Load.PickValues();          <<< OLD "Load.GetValues()"
    IndexSort(keys, values);
```

```
Load.Clear();
Load = VectorMap<int, T>(pick(keys), pick(values)); <<< OLD "VectorMap<int, T>(keys,
values)"
                                <<< why i need pick() here ??????????
}
```

I'm agree with you and I'm so sorry for the code not always in English. I wrote it year ago and now I have thousand rows to update.
Sure I'll take it in consideration for the next project!

Thanks,
Matteo
