Travaux pratiques

1. Chargement de scripts dans tiny-lisp

Le module `Labmain.cpp` définit la fonction `load_script()` utilisée pour charger un script en mémoire et lui appliquer la coloration syntaxique. Cette fonction s'appuie sur une version sécurisée de la librairie `stdlib`, et ses méthodes sont reconnaissables par leur suffixe **`_s`** (pour *secure*).

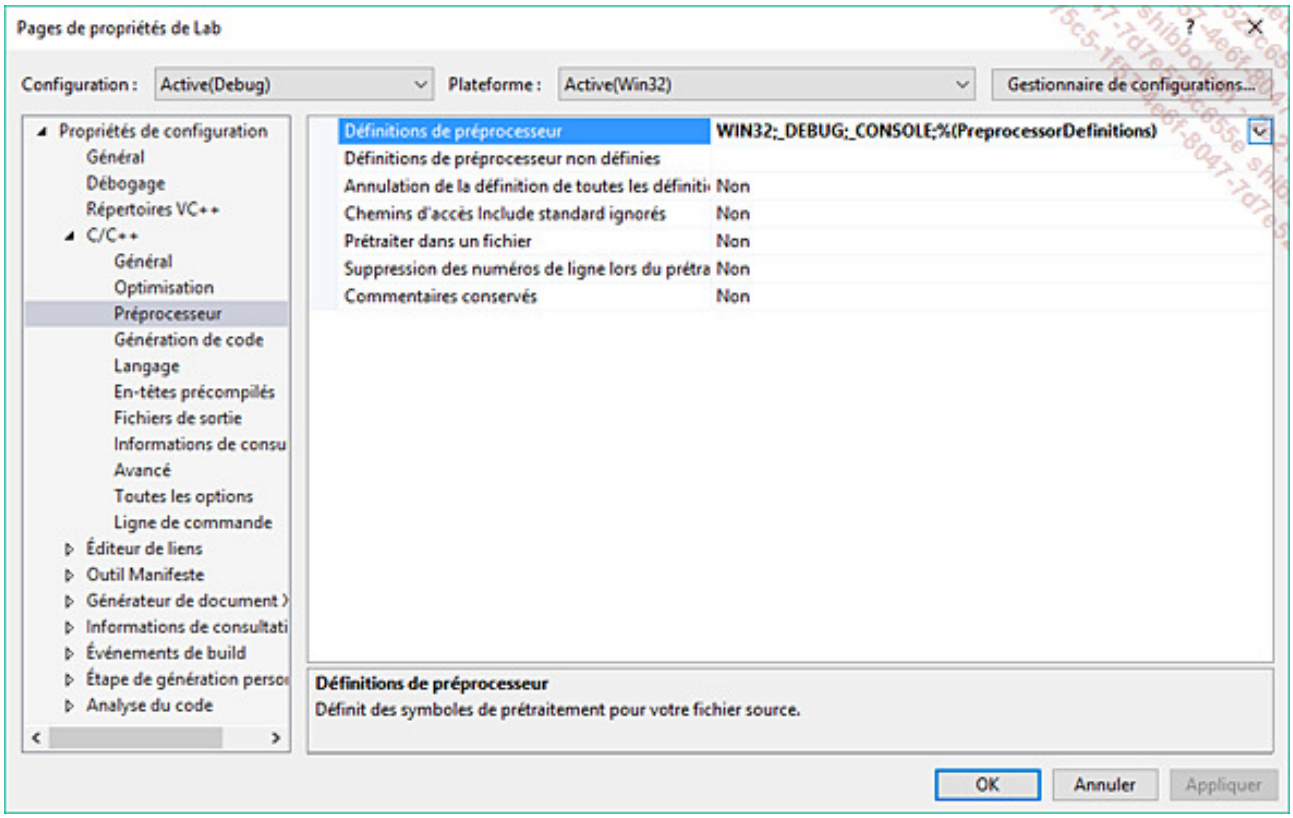
```
#pragma region load_script
void load_script(char*filename,bool show_script=false)
{
  string script;
 FILE* f = NULL;
 try
 {
   // utilise la version sûre du C++ Runtime
   int err = fopen_s(&f,(const char *)filename, "rb");
   if (err != 0)
   {
     cerr << "erreur d'ouverture de " << filename << endl;
     return;
   }

   int c;
   char buf[4001];
   while ((c = fread(buf, 1, 4000, f)) > 0)
   {
     buf[c] = 0;
     script.append((char*)buf);
   }
   fclose(f);
   f = NULL;

   if (show_script) {
     cout << ColorConsole::fg_blue << ColorConsole::bg_white;
     cout << script << endl;
```

```
    }
    // charge en consolebox
    consoleBox->new_text();
    consoleBox->set_text(script);
  }
  catch (...)
  {
    cerr << "erreur lors de la lecture du fichier" << endl;
    if(f!=NULL)
      fclose(f);
  }
}

void load_script()
{
  char filename[500];
  printf_s("Fichier ? ");
  scanf_s("%s", filename,500);

  load_script(filename,true);
}
#pragma endregion
```

2. Supprimer les erreurs liées à la librairie non sécurisée

Pour réussir avec Visual Studio 2015 la compilation d'appels de fonctions non sûres, il faut d'abord ajouter la directive **`_CRT_SECURE_NO_WARNINGS`** dans les propriétés du préprocesseur.

On édite ces directives avec la commande **Projet - Propriétés de Lab** :



Il faut ensuite utiliser la commande **Modifier** et ajouter la directive `_CRT_SECURE_NO_WARNINGS` :