```
#ifndef __iofilter_h__
#define __iofilter_h__

/* Í þessari skrá eru skilgreindir streambuf erfingjar
   sem nota má til að umbreyta straumum milli stafasetta
   með umbreytingartöflum */

#include <iostream>
#include <sstream>

namespace ff {

extern const char* _trans_861_iso;
extern const char* _trans_iso_861;

inline unsigned char translate(unsigned char c, const char* table) {
        if (c < 128) return c;
        return table[c-128];
}

inline void translatebuffer(char* buf, size_t n, const char* table) {
        for (size_t i = 0; i < n; i++) {
                unsigned char c = buf[i];
                if (c < 128) continue;
                buf[i] = table[c-128];
        }
}

class ofilterbuf : public std::streambuf {
        const char* _table;
        std::streambuf* _dst;
        char* _buffer;
public:
        ofilterbuf(const char* table, std::streambuf* dst, int buflen = 512)
                : _table(table), _dst(dst) {
                        _buffer = new char[buflen];
                        setp(_buffer, _buffer+buflen);
                }
        virtual ~ofilterbuf() {
                sync();
                delete [] _buffer;
        }

        virtual int sync();
        virtual int overflow(int ch);
};

class ifilterbuf : public std::streambuf {
        std::streambuf* _src;
        const char* _table;
        char* _buffer;
        size_t _buflen;
public:
        ifilterbuf(const char* table, std::streambuf* src, int buflen = 512)
                : _table(table), _src(src), _buflen(buflen) {
                        _buffer = new char[_buflen];
                        setg(_buffer, _buffer+_buflen, _buffer+_buflen);
                }
                virtual ~ifilterbuf() { delete [] _buffer; }

        virtual int underflow();
};

}

#endif /* __iofilter_h__ */
```

```
#include "iofilter.h"

using namespace ff;
using namespace std;

/* translate strengir byrjar á staf 128, stafir
   sem ekki finnast í target stafasetti eru settir sem
   stafurinn \137 ('_') */

const char* ff::_trans_861_iso =
        "\307\374\351\342\344\340\345\347\352\353\350\320\360\336\304"
        "\305\311\346\306\364\366\376\373\335\375\326\334\370\243\330"
        "\137\137\341\355\363\372\301\315\323\332\277\137\254\275\274"
        "\241\253\273\137\137\137\137\137\137\137\137\137\137\137\137"
        "\137\137\137\137\137\137\137\137\137\137\137\137\137\137\137"
        "\137\137\137\137\137\137\137\137\137\137\137\137\137\137\137"
        "\137\137\137\137\137\137\137\337\137\137\137\137\265\137\137"
        "\137\137\137\137\137\137\137\137\261\137\137\137\137\367\137"
        "\260\137\267\137\137\262\137\240"
;

const char* ff::_trans_iso_861 =
        "\137\137\137\137\137\137\137\137\137\137\137\137\137\137\137"
        "\137\137\137\137\137\137\137\137\137\137\137\137\137\137\137"
        "\137\137\377\255\255\234\234\234\234\234\234\234\234\256\252"
        "\252\252\252\370\361\375\375\375\346\346\372\372\372\372\257"
        "\254\253\253\250\250\244\244\244\216\217\222\200\200\220\220"
        "\220\220\245\245\245\213\213\213\246\246\246\231\231\235\235"
        "\247\247\232\227\215\341\205\240\203\203\204\206\221\207\212"
        "\202\210\211\211\241\241\241\214\214\214\242\223\223\224\366"
        "\233\233\243\226\201\230\225\225"
;

int ofilterbuf::sync() {
        streamsize n = pptr() - pbase();
        if (0 == n) return 0;
        translatebuffer(pbase(), n, _table);
        if (n != _dst->sputn(pbase(), n)) {
                cerr << "Móttökustraumur í ofilterbuf gat ekki tekið við öllu." << endl;
                return EOF;
        }
        pbump(-n);
        return 0;
}

int ofilterbuf::overflow(int ch) {
        streamsize n = pptr() - pbase();
        if (n && sync())
                return EOF;
        if (ch != EOF) {
                _dst->sputc(translate(ch, _table));
        }
        return 0;
}

int ifilterbuf::underflow() {
        if (gptr() < egptr())
                return *gptr();
        streamsize read = _src->sgetn(eback(), _buflen);
        if (0 == read && EOF == _src->sgetc()) {
                return EOF;
        }
        translatebuffer(eback(), read, _table);
        setg(eback(), eback(), eback()+read);
        return *gptr();
}
```

```
#ifndef __myast_h__
#define __myast_h__

#include <antlr/CommonAST.hpp>

namespace ff {

class ffAST;
typedef antlr::ASTRefCount<ffAST> RefffAST;

/* ffAST er klasi sem nota má sem AST hnút í ANTLR trjásmið.
   Hann sér sjálfur um að halda utan um úr hvaða línu inntaksins
   hann var smíðaður. */

class ffAST : public antlr::CommonAST {
private:
    int _line;
public:
        ffAST() : _line(0) {}
        virtual ~ffAST() {}

    int getLine() const
        {
                return _line;
        }

        void initialize(antlr::RefToken t)
        {
                antlr::CommonAST::initialize(t);
                _line = t->getLine();
        }

        void initialize(antlr::RefAST t)
        {
                antlr::CommonAST::initialize(t);
                _line = (static_cast<ffAST*>(t.get()))->_line;
        }

        static antlr::RefAST factory()
        {
                return antlr::RefAST(new ffAST);
        }

};

}

#endif
```

```
#ifndef __ff_utils_h__
#define __ff_utils_h__

#include <string>

namespace ff {

        char styriStafur(std::string takn);
        int hex2int(char c);

}

#endif
```

```
#include "utils.h"

namespace ff {

        char styriStafur(std::string takn) {
                // assert (takn[0] == '\\')
                switch (takn[1]) {
                case '$':
                        // assert (string.length() == 4)
                        return (char) (
                                          hex2int(takn[2]) << 4
                                        | hex2int(takn[3]));

                case 'c': case 'C':  return (char) 13;
                case 'b': case 'B':     return (char)  7;
                case 'e': case 'E':     return (char) 27;
                case 'f': case 'F':     return (char) 12;
                case 'n': case 'N':     return (char) 10;
                case 't': case 'T':     return (char)  9;

                case '0': case '1': case '2': case '4':
                case '5': case '6': case '7': case '8':
                case '9':
                        return takn[1] - '0';

                default:
                        return takn[1];
                }
        }

        int hex2int(char c) {
                if (c >= '0' && c <= '9')
                        return c - '0';
                else if (c >= 'a' && c <= 'f')
                        return c - 'a' + 10;
                else if (c >= 'A' && c <= 'F')
                        return c - 'A' + 10;
                else
                        return 0;
        }

}
```