M-x column-number-mode; line-number-mode; tool-bar-mode; customize-face

C-x m <function>

<cstring>:

char\* **strstr**(char\* str, const char\* target);

const char\* **strstr**(const char\* str, const char\* target);

returns \*p to 1st occurrence in str of target, else null \*p

int **strlen**(const char\*);

  returns length of string, not counting sentinel character

char\* **strcpy**(char\* dest, const char\* source);

char\* **strtok**(char\*, const char\* delims);

for(char\* p=strtok(str,".,"); p!=NULL; p=strtok(NULL,".,"))

char\* **strcat**(char\* dest, const char\* source);

int **strcmp**(const char\* str1, const char\* str2);

returns 0 if strings identical, returns >0 if 1st unmatched char has greater value in str1 (vice versa)

int **strncmp**(const char\*, const char\*, int n);

strcmp() on first n characters of both strings

<istream><fstream>:

istream.**getline**(char\*, int size);

istream.**getline**(char\*, int size, char delim);

istream.**get**(char& c); int istream.**get**();

int istream.**peek**(); returns next character to be extracted, without extracting

istream.**seekg**(int pos); istream.**seekg**(streamoff offset, int pos);

input.seekg(0, input.end); points to last char in input

input.seekg(4, input.beg); points to 5th char in input

istream.**putback**(char to\_put\_back); decrements current location in stream

<ostream><fstream>:

ostream.**put**(char c);

outstream.**open**(const char\*);

<cctype>:

int **isalnum** (int c); int **isalpha** (int c); int **isdigit** (int c);

int **islower** (int c); int **ispunct** (int c); int **isupper** (int c);

int **isspace** (int c); int **iscntrl** (int c);

a control character does not occupy a printing position on a display

int **tolower** (int c); int **toupper** (int c); these do nothing if not letter

<cstdlib>:

int **atoi**(const char\* str);

string to integer; discards whitespace until ‘-‘ or digit met

<cstdio>:

int sprintf(char\* str, const char\* format, ...);

char str1[10], str2[50];

int n = 109073, a = 5, b = 4;

sprintf(str, “%d”, n); //out: “109073”

sprintf(str2, “%d plus %d is %d”, a, b, a+b); //out: “5 plus 4 is 9”

if handling digits only:

    int digit = 6;

  char cdigit = digit + '0';

other:

**typedef** int\* iptr;

**const char\*** is same as char const\*

**char\* const** is a constant pointer to a char

**const char\* const** is a constant pointer to a const char

<cmath>:

double pow(double base, double exponent);

<vector>:

typedef **vector**<Class> Vec;

typedef vector<Class>::**iterator** VecIt;

vecobj.**push\_back**(Class element); append\_1\_element

vecobj.**insert**();

vecobj.**find**(vecobj.begin(), vecobj.end(), Class element);

returns VecIt pointing to element found

linked lists:

  struct Node {

    char word[MAX\_WORD\_LENGTH];

    Node \*ptr\_to\_next\_node;

  };

- IntPtr &ptr really important, othw confusion betw &\* and \*&

- never Node a\_list; nor Nodeptr a\_list; rather a\_list = NULL;

- always call a\_node by reference in f(), because want previous  to point to modified a\_node

- assign\_linked\_list(): finish with previous->next = NULL;

2D arrays, load map:

  char \*\***allocate\_2D\_array**(int rows, int columns) {

    char \*\*m = new char \*[rows];

    assert(m);

    for (int r=0; r<rows; r++) {

      m[r] = new char[columns];

      assert(m[r]);

    }

    return m;

  }

  void **deallocate\_2D\_array**(char \*\*m, int rows) {

    for (int r=0; r<rows; r++)

      delete [] m[r];

    delete [] m;

  }

  char \*\***load\_maze**(const char \*filename, int &height, int &width) {

    bool success = **get\_maze\_dimensions**(filename, height, width);

    if (!success)

      return NULL;

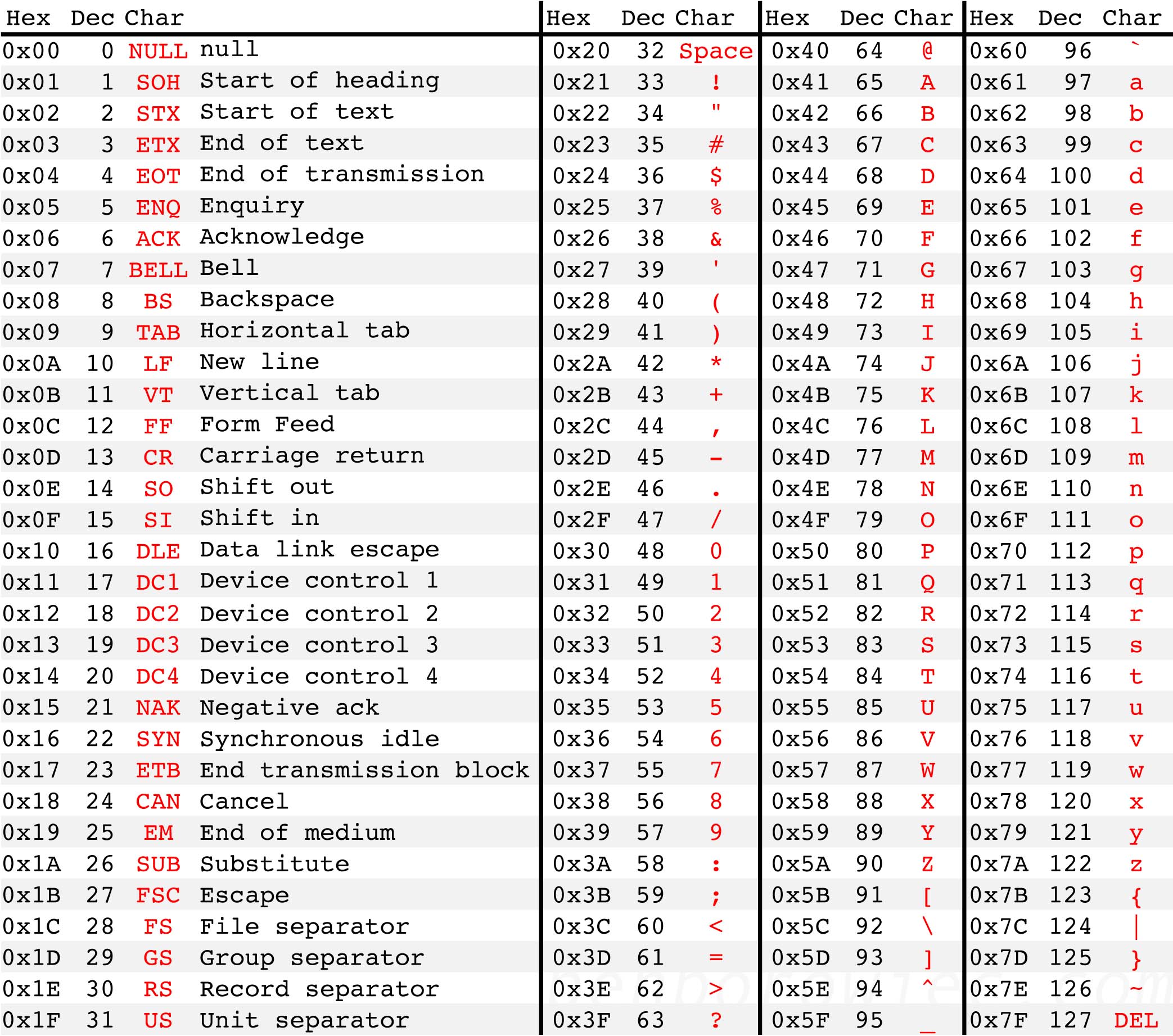
      char \*\*m = allocate\_2D\_array(height, width);

      ifstream input(filename);

    char line[512];

    for (int r = 0; r<height; r++) {

      input.getline(line, 512);



      strcpy(m[r], line);

    }

    return m;

  }

selection sort:

void **selection\_sort**(int a[], int length) {

for (int count = 0 ; count < length - 1 ; count++) swap(a[count],a[minimum\_from(a,count,length)]);

}

int **minimum\_from**(int a[], int position, int length) {

int min\_index = position;

for (int count = position + 1 ; count < length ; count ++)

if (a[count] < a[min\_index])

min\_index = count;

return min\_index;

}

void **swap**(int& first, int& second) {

int temp = first;

first = second;

second = temp;

}

silly stuff:

- ostreams and istreams as args must be called by ref

- only way to return cstring is with dynamic arrays

- 'int \*p1 = NULL, \*p2 = p1; p1=new int; \*p1=1; cout<<p2' segfault

- \_.cpp doesn't need #include"\_.h" unless a function in .cpp is

  called before having been defined

- 'using namespace std' must appear before #include"\_.h" in any file!

  otherwise include libraries in header file, and include only header

  in all other files

- palindrome q1 took you 15 mins, because you're not commenting

  as you code where you might be writing a mistake

- braille print question humiliated you with hours of work

- can't change size of dynarray. must allocate a new array, copy

  values, delete original array, change member variable to point

  to the new array.

makefile:

exam: main.cpp exam.cpp exam.h

g++ -g -Wall main.cpp exam.cpp -o exam