Day 19 Exploring the C Function Library

Mathematical Functions

pg 534-536 lists math functions

all functions return double for accuracy

Dealing with Time

time.h

2 methods for representing time

1. number seconds elapsed from midnight Jan 1, 1970 time values are stored as type long integers with typedef to time_t and clock_t
2. time broken into components- year mo day using struct tm- see pg 538 but does seconds, minutes, day of week, even flag for daylight savings

obtaining current time

current time on system clock-

```
time_t time(time_t *timeptr);
```

number seconds since 0000 jan 1 1970

so:

```
time_t now;
now = time(0);
```

or:

```
time_t now;
time_t *ptr_now = &now;
time(ptr_now);
```

coverting between time representations

time since 0000 jan 1 1970 not useful- localtime() represents time as tm struct:

```
struct tm *localtime(time_t *ptr);
```

returns pointer to tm struct- only need to declare a pointer to type tm

struct tm overwritten everytime called-if want to capture a time, must declare a separate type tm struct and copy values from that

to reverse from tm struct to type time_t value:

```
time_t mktime(struct tm *ntime);
```

back to time since 1970

displaying time

convert to formatted strings

```
char *asctime(struct tm *ptr);
char *ctime(time_t *ptr);
```

ctime type time_t asctime type tm struct

both return pointer to static, null terminated 26 character string: Thu Jun 13 19:20:22 2020

for more control over format, use strftime()

```
size_t strftime(char *s, size_t max, char *fmt, struct tm *ptr);
```

takes tm struct pointed to by ptr formats it according to format string fmt, and writes result as null terminated string to memory location pointed to by s. max specifies amount of space at s

pg 540 for conversion specifiers for strftime()

calculating time difference

can calculate difference in secs between two times with difftime()

```
double difftime(time_t later, time_t earlier);
```

commonly used calc elapsed time

clock() calculates amount of time since program began execution in 1/100 second units

```
clock_t clock(void);
```

Error Handling

assert.h

```
void assert(int expression);
```

expression anything you want to test- varaible or any C expression

if true does nothing- if false displays error message on stderr and aborts execution

compiled in debug mode.

does not solve compilation errors- rather if program itself is running incorrectly, can sprinkle in assert() to diagnose trouble

if NDEBUG is defined- asserts turn off

```
#define NDEBUG
```

ex: believe that problem is variable taking on negative value put something like

```
assert(interest_rate >= 0);
```

after the point you think it is happening

errno.h and perror()

errno.h defines several macros used to define and document runtime errors

table 19.2 pg.546 lists error constants in errno.h

if nonzero error has occurred

only need errno.h defined if you want to use symbolic constants from errno

perror() will display a user defined message when a system or library call error occurs or return "no error"

```
void perror(const char *msg);
```

msg points to optional user defined message

does not prompt program to take action- COULD have action program takes defined by testing value errno

ex of calling errno:

```
printf("errno = %d.\n", errno);
```

Searching with bsearch() and qsort()

performs binary search of data array looking for array element that matches a key

array must be sorted in ascending order

must provide comparison function

```
void *bsearch(const void *key, const void *key, const void *base, size_t num, size_t width,
```

key pointer to data item being searched for. base is pointer to first element of array being searched. num number elements in array. width size in bytes of each element. size_t refers to data type returned by the sizeof() operator which is unsigned. sizeof() operator usually used to obtain the values for num and width. cmp is a pointer to comparison function.

cmp must: 1. passed pointers to 2 data items 2. returns type int as follows: < 0 Element 1 is less than element 2. 0 Element 1 is equal to element 2. > 0 Element 1 is greater than element 2

return value is type void pointer to first array element that matches the key or null if not found. must cast returned pointer to proper type before using it

this returns num, the number of elemetns in the array:

```
sizeof(array)/sizeof(array[0])
```

starts in middle compares $>$ or $<$ then goes ascending or descending accordingly

qsort- sorts array into order stdlib.h

```
void qsort(void *base, size_t num, size_t size, int (*cmp)(const void *element1, const void
```

no return value

pg 551 for examples of using bsearch() and qsort()

binary search requires n comparisons to search array of 2^n elements

ALWAYS VALIDATE USER ENTERED DATA