Library Functions

- Calculate the hypotenuse of a triangle given two perpendicular sides
  - pow(x, y) and sqrt(x) are functions defined in math.h library

\[ Hyp = \sqrt{a^2 + b^2} \]

- Hyp = sqrt(pow(a, 2) + pow(b, 2));

User Defined Functions

```c
#include <stdio.h>
...
int myfunc(int x); /* Define function prototype */
...
int main(void)
{
    ...
    z = myfunc(4); /* Use (call) function within main or other functions */
    ...
}

int myfunc(int x) /* Define function itself */
{
    ...
    ...
}
```

Dr. Sadık Eşmelioğlu  CENG 114 – Spring 2009-10  51
User Defined Functions

• Define function prototype
  
  \[ \text{type name}(\text{argument}); \]

  where

  • type is the type of value function returns (void if nothing)
  • name is the name of the function (follow variable naming convention)
  • argument is the type and the name of the variable it is expecting (void if nothing). Can be more than one variable separated by commas.

User Defined Functions

• Call function
  
  \[ \text{name}(\text{argument}); \]

  where

  • argument is the value we pass to the calling function (void if nothing). Can be more than one variable separated by commas.
User Defined Functions

- Define function
  
  ```c
  type
  name(argument)
  {
    ...
  }
  ```

  where
  - `type` is the type of value function returns (void if nothing)
  - `name` is the name of the function (follow variable naming convention)
  - `argument` is the type and the name of the variable it is expecting (void if nothing). Can be more than one variable separated by commas.

Example 1

```c
#include <stdio.h>
#include <math.h>

double mysqr(double x);
/* Squares given argument */

int main(void)
{
  double a, b; /* input - Sides of right triangle */
  double Hyp; /* output - The Hypotenuse */

  /* Get Sides a and b */
  printf("Enter sides a and b: ");
  scanf("%lf %lf", &a, &b);

  /* Calculate Hypotenuse */
  Hyp = sqrt(mysqr(a) + mysqr(b));

  /* Display Result */
  printf("Hypotenuse of Triangle is %.0f", Hyp);

  return(0);
}

double mysqr(double x)
{
  double res;
  res = x * x;
  return(res);
}
```
Example 2

```c
#include <stdio.h>
#include <math.h>

double ScaleUp10(double x, int a); /* Multiplies x by 10 to a */
double ScaleDown10(double x, int a); /* Divides x by 10 to a */

int main(void)
{
    double x; /* input – number to be rounded */
    int a; /* # of digits to be rounded */
    double y; /* output - rounded number */

    /* Number to be rounded and # of significant digits*/
    printf("Enter Number to be rounded:");
    scanf("%lf", &x);
    printf("Enter # of digits to be rounded:");
    scanf("%d", &a);

    /* Round Number */
    y = (int) (ScaleUp10(x, a) + 0.5);
    y = ScaleDown10(y, a);

    /* Display Result*/
    printf("Rounded Number is %f ", y);
    return(0);
}
```

Some Useful Library Functions

- `rand()` in `stdlib.h`
  - returns a random number between 0 and RAND_MAX
- `srand(unsigned int)` in `stdlib.h`
  - uses the unsigned int number to initialize the random number generator
- `time (NULL)` in `time.h`
  - returns the number of seconds elapsed between current time and 01.01.1970 00:00 GMT
- `srand((unsigned) time(NULL))`
  - seeds the random number generator with a different number every time it is executed
Black Jack

/*
 * One person black jack game provides the running total
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char GetNumber(int); /* Gets the name of the card given its number */
char GetSuit(int);  /* Gets the suit 0-H, 1-S, 2-D, 3-C */
int GetValue(char);  /* Gets the face value 10, J, Q, K have value of 10 */

main(void)
{
    char Number; /* Card Name */
    char Suit;  /* Suit Name */
    char ans, junk; /* To continue or not */
    int Total = 0; /* Total value */

    srand((unsigned)time(NULL)); /* Provide a different seed number for random number generator */

    Number = GetNumber(rand() % 13);
    Suit = GetSuit(rand() % 4);

    Total += GetValue(Number);

    printf("You have %c of %c
", Number, Suit);
    printf("Your Total is %d
", Total);

    if (Total > 21) {
        printf("\n\nYou Loose\n\n");
        break;
    }

    printf("Do you want to continue? [y/n]: ");
    scanf("%c", &ans, &junk);
}

Dr. Sadık Eşmelioğlu
CENG 114 – Spring 2009-10
Black Jack

/* Given an integer from 0 – 12 returns the card name
   * 0 is 1, 1 is 2, ... , 9 is T, 10 is J, 11 is Q, 12 is K */

char GetNumber(int number)
{
    char Number;
    switch (number) {
    case 0: Number = '1'; break;
    case 1: Number = '2'; break;
    case 2: Number = '3'; break;
    case 3: Number = '4'; break;
    case 4: Number = '5'; break;
    case 5: Number = '6'; break;
    case 6: Number = '7'; break;
    case 7: Number = '8'; break;
    case 8: Number = '9'; break;
    case 9: Number = 'T'; break;
    case 10: Number = 'J'; break;
    case 11: Number = 'Q'; break;
    case 12: Number = 'K'; break;
    default: Number = '?';
    }
    return(Number);
}

Dr. Sadık Eşmelioğlu
CENG 114 – Spring 2009-10

Black Jack

/* Given an integer from 0 – 3 returns the suit name
   * 0 is H, 1 is S, 2 is D, 3 is C */

char GetSuit(int suit)
{
    char Suit;
    switch (suit) {
    case 0: Suit = 'H'; break;
    case 1: Suit = 'S'; break;
    case 2: Suit = 'D'; break;
    case 3: Suit = 'C'; break;
    default: Suit = '?';
    }
    return(Suit);
}
Black Jack

/*
 * Given card name  returns the value of card
 * card 1 - 9 has own value * T, J, Q, K have 10 value
 */

int GetValue(char Suit)
{
    int Value;
    switch (Suit) {
        case '1': Value = 1; break;
        case '2': Value = 2; break;
        case '3': Value = 3; break;
        case '4': Value = 4; break;
        case '5': Value = 5; break;
        case '6': Value = 6; break;
        case '7': Value = 7; break;
        case '8': Value = 8; break;
        case '9': Value = 9; break;
        case 'T':
        case 'J':
        case 'Q':
        case 'K': Value = 10; break;
        default: Value = -999;
    }
    return (Value);
}

Programming Project

1. Problem:
   • Write a program that asks the user to provide a number, then tells the user if this number is evenly divisible by 2, 3, and 5

2. Analysis
   • The remainder operation (%) can be used to determine if a number is evenly divisible by another number
   • If the number is 66:
     • 66 % 2 = 0 which means it is evenly divisible by 2
     • 66 % 3 = 0 which means it is evenly divisible by 3
     • etc.
Programming Project

3. Design:
   a) Prompt the user for input – Number
   b) Calculate Number % 2 – If no remainder DivBy2 is True
   c) Calculate Number % 3 – If no remainder DivBy3 is True
   d) Calculate Number % 5 – If no remainder DivBy5 is True
   e) Print the results
   • Since b, c and d are repetitive, write a function that takes number and divider, returns 1 if divisible and 0 if not

4. Coding

```c
#include <stdio.h>

int DivBy(int x, int a); /* if x is divisible by a returns 1 if not returns 0 */

int main(void)
{
    int Number; /* Input number */
    int DivBy2, /* Divisible by 2 True:1 – False:0 */
    DivBy3, /* Divisible by 3 True:1 – False:0 */
    DivBy5; /* Divisible by 5 True:1 – False:0 */
```
programming project

/* get the number to be checked */

printf("This program finds if a number is divisible by 2, 3, and 5\n");
printf("Please enter a number: ");
scanf("%d", &Number);

/* check if divisible by 2, 3, and 5 */

DivBy2 = DivBy(Number, 2);
DivBy3 = DivBy(Number, 3);
DivBy5 = DivBy(Number, 5);

/* print the outcome */

switch (DivBy2 + DivBy3 + DivBy5) {
    case 0:
        printf("%d is not divisible by 2 3 5\n", Number);
        break;
    default:
        printf("%d is divisible by", Number);
        if (DivBy2 == 1) printf(" 2");
        if (DivBy3 == 1) printf(" 3");
        if (DivBy5 == 1) printf(" 5");
}

return(0);
}
Programming Project

```c
/*
 * This function checks if first argument is divisible by second
 * Returns 1 if true and 0 if false
 */

int DivBy(int x, int a)
{
    if((x%a) == 0) return(1);
    else return(0);
}
```

5. Test:
   a) Try 13 – Expected result: None
   b) Try 66 – Expected result: 2 3
   c) Try 25 – Expected result: 5
   d) Try 60 – Expected result: 2 3 5

6. Maintenance:
   Try Adding divisibility by 4
Refactoring

- **Refactoring** is the process of changing a program’s code without changing its functionality, in order to improve *quality attributes* of the software such as:
  - readability,
  - maintainability,
  - performance,
  - extensibility,
  - portability,
  - etc.

Using Input/Output Files

- Declare file pointers
  ```c
  FILE *inpf, /* pointer to a file for input */
  *outpf; /* pointer to a file for output */
  ```
- Open files for reading and writing
  ```c
  inpf = fopen("data.txt", "r");
  outpf = fopen("out.txt", "w");
  ```
- Read from and Write into files
  ```c
  fscanf(inpf, "%lf", &Radius);
  fprintf(outpf, "Area of Circle is %f\n", AreaCircle);
  ```
- Close all open files
  ```c
  fclose(inpf);
  fclose(outpf);
  ```
scanf statement

- `scanf("format string", input list)`
- `fscanf(file pointer, "format string", input list)`
  - file pointer
  - pointer to a file
  - format string
    - is always enclosed in double quotes ""
    - contains a place holder for each variable in the input list
    - place holders are `%c` (char), `%d` (int), `%f` (double, float)
  - input list
    - is a comma separated list of variables to be read into
    - these variable should have an `&` sign in front of them
- returns
  - the number of variables read and assigned
  - a negative number if EOF is read

Grades

/*
* Reads grades from an input file - Grades.txt
* Each line contains 3 grades, Midterm 30%, Lab 30%, and Final 40%
* Calculates Term Grades both in Number and letter format
* 90-100 is AA, 85-89 is BA, etc.
* the output is placed in another file called GradesOut.txt
*/

#include <stdio.h>  /* Defines fopen, fclose, scanf, fprintf, and EOF */
#define MID_PERC 0.3  /* Midterm Percentage */
#define LAB_PERC 0.3  /* Lab Percentage */
#define FIN_PERC 0.4  /* Final Percentage */
Grades

int main(void)
{
    FILE *gradesin, /* Input file */
        *gradesout; /* Output file */
    int MidGrd, /* Midterm Grade read from file */
        LabGrd, /* Lab Grade read from file */
        FinGrd, /* Final Grade read from file */
        ReadStat; /* Status returned by fscanf */
    double NumGrd; /* Calculated numerical grade */
    int NumGrdR; /* Rounded numerical grade */
    char LtrGrd1, /* Left side of letter grade */
        LtrGrd2; /* right side of letter grade */

    /* Open files */
    gradesin = fopen("Grades.txt", "r");
    gradesout = fopen("GradesOut.txt", "w");

    /* Read First Line */
    ReadStat = fscanf(gradesin, "%d %d %d", &MidGrd, &LabGrd, &FinGrd);

    while (ReadStat != EOF) { /* When EOF is reached exit */
        /* Calculate End Term Grade */
        NumGrd = MidGrd * MID_PERC + LabGrd * LAB_PERC + FinGrd * FIN_PERC;

        /* Round Grade up */
        NumGrdR = (int)(NumGrd + 0.5);
    }
/* Find Letter equivalent grade */
if (NumGrdR >= 90)
    {LtrGrd1 = 'A'; LtrGrd2 = 'A';}
else if (NumGrdR >= 85)
    {LtrGrd1 = 'B'; LtrGrd2 = 'A';}
else if (NumGrdR >= 80)
    {LtrGrd1 = 'B'; LtrGrd2 = 'B';}
else if (NumGrdR >= 75)
    {LtrGrd1 = 'C'; LtrGrd2 = 'B';}
else if (NumGrdR >= 70)
    {LtrGrd1 = 'D'; LtrGrd2 = 'C';}
else if (NumGrdR >= 65)
    {LtrGrd1 = 'D'; LtrGrd2 = 'D';}
else if (NumGrdR >= 60)
    {LtrGrd1 = 'F'; LtrGrd2 = 'F';}

/* Print Numerical and Letter Grades into output file */
fprintf(gradesout, "%.1f	%c%c
", NumGrd, LtrGrd1, LtrGrd2);

/* Read Next Line */
ReadStat = fscanf(gradesin, "%d %d %d", &MidGrd, &LabGrd, &FinGrd);
}
/* Close input and output files */
fclose(gradesin);
fclose(gradesout);
return (0);

---

**Functions – Input/Output**

- **void func( void)**
  - No input
  - No return
  - No output

- **void func( int x)**
  - One input – x
  - No return
  - No output

- **int func( void)**
  - No input
  - Returns int
  - One output – returned int

- **int func( int x)**
  - One input – x
  - Returns int
  - One output – returned int

- **int func( int x, int y)**
  - Two inputs – x and y
  - Returns int
  - One output – returned int
Functions with Output Param

- Prototype of a function with one input and two output parameters to be declared before the main function
  
  ```c
  type func (type inp1, type *outp1, type *outp2);
  ```

- Definition to be made after the main function
  
  ```c
  type
  func (type inp1, type *outp1, type *outp2)
  {
    :
    :
  }
  ```

- Call within main
  
  ```c
  func(w, &x, &y);
  ```

---

### Main Function

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>7000</td>
<td>-2.1</td>
</tr>
<tr>
<td>x</td>
<td>7001</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>7002</td>
<td>0</td>
</tr>
</tbody>
</table>

```c
func(w, &x, &y);
```

### User Function

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>7910</td>
<td>-2.1</td>
</tr>
<tr>
<td>p</td>
<td>7911</td>
<td>7001</td>
</tr>
<tr>
<td>q</td>
<td>7912</td>
<td>7002</td>
</tr>
</tbody>
</table>

```c
int func (double a, int *p, int *q);
```
Example

/*
 * Demonstrates the use of a function with input and output parameters.
 */

#include <stdio.h>
#include <math.h>

void separate(double num, char *signp, int *wholep, double *fracp);

int main(void)
{
    double value;            /* input - number to analyze */
    char    sn;              /* output - sign of value */
    int     whl;             /* output - whole number magnitude of value */
    double  fr;              /* output - fractional part of value */

    /* Gets data */
    printf("Enter a value to analyze> ");
    scanf("%lf", &value);

    /* Separates data value into three parts */
    separate(value, &sn, &whl, &fr);

    /* Prints results */
    printf("Parts of %.4fn sign: %c\n", value, sn);
    printf(" whole number magnitude of value: %dn\n", whl);
    printf(" fractional part:  %.4fn\n", fr);

    return (0);
}
Example

```c
/*
 * Separates a number into three parts: a sign (+, -, or blank),
 * a whole number magnitude, and a fractional part.
 */

void separate(double num, char *signp, int *wholep, double *fracp)
{
    double magnitude; /* local variable - magnitude of num */

    /* Determines sign of num */
    if (num < 0) *signp = '-';
    else if (num == 0) *signp = ' ';
    else *signp = '+';

    /* Separate num into whole (no sign) and fractional parts */
    magnitude = fabs(num);
    *wholep = floor(magnitude);
    *fracp = magnitude - *wholep;
}
```

Arrays

- Arrays are variables that allow storing multiple data that are related with each other.
- Arrays, similar to single value variables, can be of type int, char, double, etc.
- Let us represent temperature measurements taken every hour in a day:
  - \( T_0, T_1, \ldots, T_{23} \)
  - Array representation \( T[0], T[1], \ldots, T[23] \)
  - The numbers in brackets are called subscripts
  - The array has 24 elements, i.e. the array size is 24 (\( N = 24 \))
  - \( T_i \) where \( i \) goes from 0 to \( N - 1 \)