Introduction to Scientific and Engineering Computation (BIL 104E)

Lab 11

Applying Pointers: Pointer Aritmetic

```
#include <stdio.h>
main()
{
   char *ptr_ch;
   int *ptr_int;
   double *ptr_db;
/* char pointer ptr_ch */
   printf("Current position of ptr_ch: %p\n", ptr_ch);
   printf("The position after ptr_ch + 1: %p\n", ptr_ch + 1);
   printf("The position after ptr_ch + 2: %p\n", ptr_ch + 2);
   printf("The position after ptr_ch - 1: %p\n", ptr_ch - 1);
   printf("The position after ptr_ch - 2: %p\n", ptr_ch - 2);
/* int pointer ptr_int */
   printf("Current position of ptr_int: %p\n", ptr_int);
   printf("The position after ptr_int + 1: %p\n", ptr_int + 1);
   printf("The position after ptr_int + 2: %p\n", ptr_int + 2);
   printf("The position after ptr_int - 1: %p\n", ptr_int - 1);
   printf("The position after ptr_int - 2: %p\n", ptr_int - 2);
```

Applying Pointers: Pointer Aritmetic

```
/* double pointer ptr_ch */
   printf("Current position of ptr_db: %p\n", ptr_db);
   printf("The position after ptr_db + 1: %p\n", ptr_db + 1);
   printf("The position after ptr_db + 2: %p\n", ptr_db + 2);
   printf("The position after ptr_db - 1: %p\n", ptr_db - 1);
   printf("The position after ptr_db - 2: %p\n", ptr_db - 2);
   getchar();

   return 0;
}
```

Applying Pointers: Pointer Aritmetic

Computer Screen

```
Current position of ptr_ch: 76A8B0FF
The position after ptr_ch + 1: 76A8B100
The position after ptr_ch + 2: 76A8B101
The position after ptr_ch - 1: 76A8B0FE
The position after ptr_ch - 2: 76A8B0FD
Current position of ptr_int: 0022FF48
The position after ptr_int + 1: 0022FF4C
The position after ptr_int + 2: 0022FF50
The position after ptr_int - 1: 0022FF44
The position after ptr_int - 2: 0022FF40
Current position of ptr_db: 00000002
The position after ptr_db + 1: 0000000A
The position after ptr_db + 2: 00000012
The position after ptr_db - 1: FFFFFFFA
The position after ptr_db - 2: FFFFFFF2
```

Applying Pointers: Pointer Aritmetic

```
#include<stdio.h>

main()
{
   int  *ptr_array;
   int  my_array[]= {1, 2, 3, 4, 5};
   int i;

   ptr_array = &my_array[0];

   for(i = 0; i <= 4 ; i++) {
   printf("address via array = %p,  address via pointer =  %p\n", &my_array[i], ptr_array + i);
   printf("the my_array[%d] via pointer is %d\n ", i, *(ptr_array + i));
   }
   getchar();
   return 0;
}
```

Applying Pointers: Pointer Aritmetic

Computer Screen

```
address via array = 0022FF10,  address via pointer =  0022FF10
the my_array[0] via pointer is 1
 address via array = 0022FF14,  address via pointer =  0022FF14
the my_array[1] via pointer is 2
 address via array = 0022FF18,  address via pointer =  0022FF18
the my_array[2] via pointer is 3
 address via array = 0022FF1C,  address via pointer =  0022FF1C
the my_array[3] via pointer is 4
 address via array = 0022FF20,  address via pointer =  0022FF20
the my_array[4] via pointer is 5
```

Applying Pointers: Passing Arrays to Functions

Passing Arrays to Functions

```
1:  /* 16L04.c: Passing arrays to functions */
2:  #include <stdio.h>
3:
4:  int AddThree(int list[]);
5:
6:  main()
7:  {
8:     int sum, list[3];
9:
10:    printf("Enter three integers separated by spaces:\n");
11:    scanf("%d%d%d", &list[0], &list[1], &list[2]);
12:    sum = AddThree(list);
13:    printf("The sum of the three integers is: %d\n", sum);
14:
15:    return 0;
16: }
17:
18: int AddThree(int list[])
19: {
20:    int i;
21:    int result = 0;
22:
23:    for (i=0; i<3; i++)
24:       result += list[i];
25:    return result;
26: }
```

Computer Screen

```
Enter three integers separated by spaces:
10  20  30
The sum of the three integers is: 60
```

Applying Pointers: Arrays of Pointers

```
#include<stdio.h>
main()
{
   int  *my_array[3];
   int i, x, y, z;

   x = 1;
   y = 2;
   z = 3;
   printf("x = %d, y = %d , z = %d\n", x, y, z);
   my_array[0] = &x;
   my_array[1] = &y;
   my_array[2] = &z;
   for(i = 0; i <= 2 ; i++)
        *my_array[i] = 10 *(i + 1);
   printf("x = %d, y = %d , z = %d\n", *(my_array[0]), *(my_array[1]), *(my_array[2]));

   getchar();
   return 0;
}
```

Applying Pointers: Arrays of Pointers

Computer Screen

```
x = 1, y = 2 , z = 3
x = 10, y = 20 , z = 30
```

Allocating Memory : The malloc() Function

Using the `malloc()` Function

```
1:  /* 17L01.c: Using the malloc function */
2:  #include <stdio.h>
3:  #include <stdlib.h>
4:  #include <string.h>
5:  /* function declaration */
6:  void StrCopy(char *str1, char *str2);
7:  /* main() function */
8:  main()
9:  {
10:    char str[] = "Use malloc() to allocate memory.";
11:    char *ptr_str;
12:    int result;
13:    /* call malloc() */
14:    ptr_str = malloc( strlen(str) + 1);
15:    if (ptr_str != NULL){
16:       StrCopy(str, ptr_str);
17:       printf("The string pointed to by ptr_str is:\n%s\n",
18:              ptr_str);
19:       result = 0;
20:    }
21:    else{
22:       printf("malloc() function failed.\n");
23:       result = 1;
24:    }
25:    return result;
26: }
```

Allocating Memory : The malloc() Function

```
26: }
27: /* function definition */
28: void StrCopy(char *str1, char *str2)
29: {
30:    int i;
31:
32:    for (i=0; str1[i]; i++)
33:       str2[i] = str1[i];
34:    str2[i] = '\0';
35: }
```

Computer Screen

```
The string pointed to by ptr_str is:
Use malloc() to allocate memory.
```

Releasing Allocated Memory with free()

Using the `free()` and `malloc()` Functions Together

```
1:  /* 17L02.c: Using the free() function */
2:  #include <stdio.h>
3:  #include <stdlib.h>
4:  /* function declarations */
5:  void DataMultiply(int max, int *ptr);
6:  void TablePrint(int max, int *ptr);
7:  /* main() function */
8:  main()
9:  {
10:    int *ptr_int, max;
11:    int termination;
12:    char key = 'c';
13:
14:    max = 0;
15:    termination = 0;
16:    while (key != 'x'){
17:       printf("Enter a single digit number:\n");
18:       scanf("%d", &max);
19:
20:       ptr_int = malloc(max * max * sizeof(int));  /* call malloc() */
21:       if (ptr_int != NULL){
22:          DataMultiply(max, ptr_int);
23:          TablePrint(max, ptr_int);
24:          free(ptr_int);
25:       }
26:       else{
27:          printf("malloc() function failed.\n");
28:          termination = 1;
29:          key = 'x';  /* stop while loop */
30:       }
31:       printf("\n\nPress x key to quit; other key to continue.\n");
32:       scanf("%s", &key);
33:    }
34:    printf("\nBye!\n");
35:    return termination;
36: }
```

Releasing Allocated Memory with free()

```
37: /* function definition */
38: void DataMultiply(int max, int *ptr)
39: {
40:    int i, j;
41:
42:    for (i=0; i<max; i++)
43:       for (j=0; j<max; j++)
44:          *(ptr + i * max + j) = (i+1) * (j+1);
45: }
46: /* function definition */
47: void TablePrint(int max, int *ptr)
48: {
49:    int i, j;
50:
51:    printf("The multiplication table of %d is:\n",
52:            max);
53:    printf(" ");
54:    for (i=0; i<max; i++)
55:       printf("%4d", i+1);
56:    printf("\n ");
57:    for (i=0; i<max; i++)
58:       printf("----", i+1);
59:    for (i=0; i<max; i++){
60:       printf("\n%d¦", i+1);
61:       for (j=0; j<max; j++)
62:         printf("%3d ", *(ptr + i * max + j));
63:    }
64: }
```

Releasing Allocated Memory with free()

Computer Screen

```
Enter a single digit number:
4
The multiplication table of 4 is:
   1   2   3   4
 --------------
1¦ 1   2   3   4
2¦ 2   4   6   8
3¦ 3   6   9  12
4¦ 4   8  12  16
Press x key to quit; other key to continue.
C
Enter a single digit number:
2
The multiplication table of 2 is:
   1   2
 -------
1¦ 1   2
2¦ 2   4
Press x key to quit; other key to continue.
x
Bye!
```

Allocating Memory : The calloc() Function

Using the `calloc()` Function

```
1:  /* 17L03.c: Using the calloc() function */
2:  #include <stdio.h>
3:  #include <stdlib.h>
4:  /* main() function */
5:  main()
6:  {
7:     float *ptr1, *ptr2;
8:     int i, n;
9:     int termination = 1;
10:
11:    n = 5;
12:    ptr1 = calloc(n, sizeof(float));
13:    ptr2 = malloc(n * sizeof(float));
14:    if (ptr1 == NULL)
15:       printf("malloc() failed.\n");
16:    else if (ptr2 == NULL)
17:       printf("calloc() failed.\n");
18:    else {
19:       for (i=0; i<n; i++)
20:          printf("ptr1[%d]=%5.2f,  ptr2[%d]=%5.2f\n",
21:           i, *(ptr1 + i), i, *(ptr2 + i));
22:       free(ptr1);
23:       free(ptr2);
24:       termination = 0;
25:    }
26:    return termination;
27: }
```

Allocating Memory : The calloc() Function

Computer Screen

```
ptr1[0] = 0.00,  ptr2[0] = 7042.23
ptr1[1] = 0.00,  ptr2[1] = 1427.00
ptr1[2] = 0.00,  ptr2[2] = 2787.14
ptr1[3] = 0.00,  ptr2[3] =    0.00
ptr1[4] = 0.00,  ptr2[4] = 5834.73
```

Allocating Memory : The realloc() Function

Using the realloc() Function

```
1:  /* 17L04.c: Using the realloc() function */
2:  #include <stdio.h>
3:  #include <stdlib.h>
4:  #include <string.h>
5:  /* function declaration */
6:  void StrCopy(char *str1, char *str2);
7:  /* main() function */
8:  main()
9:  {
10:    char *str[4] = {"There's music in the sighing of a reed;",
11:                    "There's music in the gushing of a rill;",
12:                    "There's music in all things if men had ears;",
13:                    "There earth is but an echo of the spheres.\n"
14:                   };
15:    char *ptr;
16:    int i;
17:
18:    int termination = 0;
19:
20:    ptr = malloc((strlen(str[0]) + 1) * sizeof(char));
21:    if (ptr == NULL){
22:      printf("malloc() failed.\n");
23:      termination = 1;
24:    }
```

Allocating Memory : The realloc() Function

```
25:     else{
26:       StrCopy(str[0], ptr);
27:       printf("%s\n", ptr);
28:       for (i=1; i<4; i++){
29:         ptr = realloc(ptr, (strlen(str[i]) + 1) * sizeof(char));
30:         if (ptr == NULL){
31:           printf("realloc() failed.\n");
32:           termination = 1;
33:           i = 4;    /* break the fro loop */
34:         }
35:         else{
36:           StrCopy(str[i], ptr);
37:           printf("%s\n", ptr);
38:         }
39:       }
40:     }
41:     free(ptr);
42:     return termination;
43: }
44: /* funciton definition */
45: void StrCopy(char *str1, char *str2)
46: {
47:     int i;
48:
49:     for (i=0; str1[i]; i++)
50:        str2[i] = str1[i];
51:     str2[i] = '\0';
52: }
```

Allocating Memory : The realloc() Function

Computer Screen

```
There's music in the sighing of a reed;
There's music in the gushing of a rill;
There's music in all things if men had ears;
There earth is but an echo of the spheres.
```

Special Data Types : Assigning Values to enum Names

Defining enum Data Types

```
1:  /* 18L01.c: Defining enum data types */
2:  #include <stdio.h>
3:  /* main() function */
4:  main()
5:  {
6:     enum language {human=100,
7:                    animal=50,
8:                    computer};
9:     enum days{SUN,
10:              MON,
11:              TUE,
12:              WED,
13:              THU,
14:              FRI,
15:              SAT};
16:
17:    printf("human: %d,  animal: %d,  computer: %d\n",
18:       human, animal, computer);
19:    printf("SUN: %d\n", SUN);
20:    printf("MON: %d\n", MON);
21:    printf("TUE: %d\n", TUE);
22:    printf("WED: %d\n", WED);
23:    printf("THU: %d\n", THU);
24:    printf("FRI: %d\n", FRI);
25:    printf("SAT: %d\n", SAT);
26:
27:    return 0;
28: }
```

Special Data Types : Assigning Values to enum Names

Computer Screen

```
human: 100,  animal: 50,  computer: 51
SUN: 0
MON: 1
TUE: 2
WED: 3
THU: 4
FRI: 5
SAT: 6
```

Special Data Types : Assigning Values to enum Names

Using the enum Data Type

```
1:  /* 18L02.c: Using the enum data type */
2:  #include <stdio.h>
3:  /* main() function */
4:  main()
5:  {
6:     enum units{penny = 1,
7:                nickel = 5,
8:                dime = 10,
9:                quarter = 25,
10:               dollar = 100};
11:    int money_units[5] = {
12:               dollar,
13:               quarter,
14:               dime,
15:               nickel,
16:               penny};
17:    char *unit_name[5] = {
18:              "dollar(s)",
19:              "quarter(s)",
20:              "dime(s)",
21:              "nickel(s)",
22:              "penny(s)"};
23:    int cent, tmp, i;
24:
```

```
25:    printf("Enter a monetary value in cents:\n");
26:    scanf("%d", &cent);  /* get input from the user */
27:    printf("Which is equivalent to:\n");
28:    tmp = 0;
29:    for (i=0; i<5; i++){
30:       tmp = cent / money_units[i];
31:       cent -= tmp * money_units[i];
32:       if (tmp)
33:         printf("%d %s ", tmp, unit_name[i]);
34:    }
35:    printf("\n");
36:    return 0;
37: }
```

Computer Screen

```
Enter a monetary value in cents:
141
Which is equivalent to:
1 dollar(s) 1 quarter(s) 1 dime(s) 1 nickel(s) 1 penny(s)
```

Special Data Types : Why Use typedef?

Using typedef Definitions

```
1:  /* 18L03.c: Using typedef definitions */
2:  #include <stdio.h>
3:  #include <stdlib.h>
4:  #include <string.h>
5:
6:  enum constants{ITEM_NUM = 3,
7:                  DELT='a'-'A'};
8:  typedef char *STRING[ITEM_NUM];
9:  typedef char *PTR_STR;
10: typedef char CHAR;
11: typedef int INTEGER;
12:
13: void Convert2Upper(PTR_STR str1, PTR_STR str2);
14:
15: main()
16: {
17:    STRING str;
18:    STRING moon = {"Whatever we wear",
19:                   "we become beautiful",
20:                   "moon viewing!"};
21:    INTEGER i;
22:    INTEGER term = 0;
23:
24:    for (i=0; i<ITEM_NUM; i++){
25:      str[i] = malloc((strlen(moon[i])+1) * sizeof(CHAR));
26:      if (str[i] == NULL){
27:        printf("malloc() failed.\n");
28:        term = 1;
29:        i = ITEM_NUM;  /* break the for loop */
30:      }
31:      Convert2Upper(moon[i], str[i]);
32:      printf("%s\n", moon[i]);
33:    }
34:    for (i=0; i<ITEM_NUM; i++){
35:      printf("\n%s", str[i]);
36:      free (str[i]);
37:    }
38:    printf("\n");
39:    return term;
40: }
41: /* function definition */
42: void Convert2Upper(PTR_STR str1, PTR_STR str2)
43: {
44:    INTEGER i;
45:
46:    for (i=0; str1[i]; i++){
47:      if ((str1[i] >= 'a') &&
48:          (str1[i] <= 'z'))
49:        str2[i] = str1[i] - DELT;
50:      else
51:        str2[i] = str1[i];
52:    }
53:    str2[i] = '\0';  /* add null character */
54: }
```

Special Data Types : Why Use typedef?

Computer Screen

Recursive Functions

Calling a Recursive Function

```
 1:  /* 18L04.c: Calling a recursive function */
 2:  #include <stdio.h>
 3:
 4:  enum con{MIN_NUM = 0,
 5:           MAX_NUM = 100};
 6:
 7:  int fRecur(int n);
 8:
 9:  main()
10: {
11:    int i, sum1, sum2;
12:
13:    sum1 = sum2 = 0;
14:    for (i=1; i<=MAX_NUM; i++)
15:      sum1 += i;
16:    printf("The value of sum1 is %d.\n", sum1);
17:    sum2 = fRecur(MAX_NUM);
18:    printf("The value returned by fRecur() is %d.\n", sum2);
19:
20:    return 0;
21: }
22: /* function definition */
23: int fRecur(int n)
24: {
25:    if (n == MIN_NUM)
26:      return 0;
27:    return  fRecur(n - 1) + n;
28: }
```

Computer Screen

```
The value of sum1 is 5050.
The value returned by fRecur() is 5050.
```

Command-Line Arguments

```
#include <stdio.h>
main(int argc, char *argv[])
{
   int i;

   printf("The value received by argc is %d.\n", argc);
   printf("There are %d command-line arguments passed to main().\n", argc);
   if(argc) {
         printf("The first command-line argument is: %s\n", argv[0]);
         printf("The rest of the command-line arguments are:\n");
         for (i=1; i<argc; i++)
            printf("%s\n", argv[i]);
   }
   return 0;
}
```

Command-Line Arguments

Computer Screen

```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\Murat ŞİMŞEK>cd C:\Dev-Cpp\Examples\lecture_code

C:\Dev-Cpp\Examples\lecture_code> main_argument.exe hello world
The value received by argc is 3.
There are 3 command-line arguments passed to main().
The first command-line argument is: main_argument.exe
The rest of the command-line arguments are:
hello
world
```