C Programming in Middle School

G. V. V. Sharma
Associate Professor,
Department of Electrical Engineering,
IIT Hyderabad

About this Book

This book introduces C programming for school children in middle school based on NCERT mathematics textbooks from Class 7 onwards.

There is no copyright, so readers are free to print and share.

April 14, 2025

Github: https://github.com/gadepall/algebra
License: https://creativecommons.org/licenses/by-sa/3.0/
and
https://www.gnu.org/licenses/fdl-1.3.en.html

Contents

1 Arithmetic

1.1 Addition and Subtraction

1.1.1 Do the following addition through a C program

$$17 + 23$$

Solution:

```
//Code by GVV Sharma
//Adding two integers
//April 14, 2025
#include <stdio.h>

//begin main function
int main(void)
{
//Declaring integers
int a = 17, b = 23;
//printing the sum
printf("%d\n",a+b);
        return 0;
}
//end main function
```

1.1.2 Do the following subtraction through a C program

$$7 - 9$$

Solution:

```
//Code by GVV Sharma
//Adding negative integer
//April 14, 2025
#include <stdio.h>

//begin main function
int main(void)
{
//Declaring integers
int a = 7, b = 9;
//printing the difference
printf("%d\n",a-b);
        return 0;
}
//end main function
```

1.1.3 Mulitply the following through a C program

$$4 \times (-8)$$

Solution:

```
//Code by GVV Sharma
//April 14, 2025
//Multiplication of numbers
#include <stdio.h>

int main(void)
{
int a = 4, b = −8;
printf(”%d\n”,a*b);
        return 0;
}
```

Compute the following

1.1.4 $(-75) + 18$
1.1.5 $19 + (-25)$
1.1.6 $27 + (-27)$
1.1.7 $(-20) + 0$
1.1.8 $(-35) + (-10)$
1.1.9 $(-10) + 3$
1.1.10 $17 - (-21)$
1.1.11 $8 \times (-2)$
1.1.12 $3 \times (-7)$
1.1.13 $10 \times (-1)$
1.1.14 $6 \times (-19)$
1.1.15 $12 \times (-32)$
1.1.16 $7 \times (-22)$
1.1.17 $15 \times (-16)$
1.1.18 $21 \times (-32)$
1.1.19 $(-42) \times 12$
1.1.20 $(-55) \times 15$
1.1.21 $(-5) \times (-6)$
1.1.22 $(-6) \times (-7)$
1.1.23 $3 \times (-1)$
1.1.24 $(-1) \times 225$
1.1.25 $(-21) \times (-30)$
1.1.26 $(-316) \times (-1)$

2 Programming

In a quiz, team A scored $a_1 = -40, a_2 = 10, a_3 = 0$ and team B scored $b_1 = 10, b_2 = 0, b_3 = -40$ in three successive rounds.

2.1 If the total scores are

$$a = a_1 + a_2 + a_3 \tag{2.0.1.1}$$

$$b = b_1 + b_2 + b_3 \tag{2.0.1.2}$$

which team scored more?
Solution:

```
//Code by Harini
//February 23, 2025
//Revised by GVV Sharma
//April 14, 2025
//add two sets of numbers and compare
#include <stdio.h>

//begin main function
```

```
int main() {
// first team scores
 int a1=−40,a2=10,a3=0;
// second team scores
 int b1=10,b2=0,b3=−40;

 //declaring scores variables
int a,b;
 //sum of scores
 a=a1+a2+a3;
 b=b1+b2+b3;
 //comparing scores
 if (a>b){
          printf("a scored more\n");
         }
 else if (a<b){
          printf("b scored more\n");
         }
 else {
          printf("they are equal\n");
}
//end comparison
 return 0;
}
//end main function
```

2.2 Write a function to compare the final scores. Check for the cases when $a = -40, b = -40; a = 30, b = 20; a = -20, b = -10$.

Solution:

```
//code by harini
//feb 23 2025
//code by GVV Sharma
//April 14 2025
//function to compare two numbers

#include <stdio.h>

//function to compare the numbers a and b
void compare(int a,int b){
          if (a>b){
          printf("a scored more\n");
}
 else if (a<b){
          printf("b scored more\n");
}
```

```
 else {
            printf(”they␣are␣equal\n”);
}
            }
//end function to compare the numbers a and b
//begin main function
int main() {
 int a=−40,b=−40;

 //call the function to compare the numbers
 compare(a,b);

 return 0;
}
//end main function
```

2.3 Use arrays and a for loop to evaluate

$$a = \sum_{i=0}^{2} a_i \tag{2.0.3.1}$$

$$b = \sum_{i=0}^{2} b_i \tag{2.0.3.2}$$

Solution:

```
//code by harini
///feb 23 2025
//revise by GVV Sharma
//April 14 2025
//compares sum of 2 arrays using a for loop
#include <stdio.h>

//compare function
void compare(int a,int b){
            if (a>b){
            printf(”a␣scored␣more\n”);
}
 else if (a<b){
            printf(”b␣scored␣more\n”);
}
 else {
            printf(”they␣are␣equal\n”);
}
            }
///end compare function
```

```
//begin main function
int main() {
            //Declaring arrays
int a1[]={−40,10,0};
int b1[]={10,0,−40};
//Initializing sums
int a=0,b=0;
   for (int i = 0; i <= 2; i++){
               a=a+a1[i];
               b=b+b1[i];
   }
//Call compare function
   compare(a,b);
 return 0;
}
//end main function
```

2.4 Revise the above code using only functions.
Solution:

```
//code by harini
//feb 23 2025
//revise by GVV Sharma
//April 14 2025
//using functions for arrays
#include <stdio.h>

//Declaring functions
void compare(int a,int b);
int sum(int a[]);

//begin main function
int main() {
            //Declaring arrays
int a1[]={−40,10,0};
int b1[]={10,0,−40};
//Initializing sums
int a=0,b=0;
//finding sum for A
a = sum(a1);
//finding sum for B
b = sum(b1);
//Call compare function
   compare(a,b);
 return 0;
}
```

```
//end main function

//compare function
void compare(int a,int b){
            if (a>b){
            printf("a scored more\n");
}
 else if (a<b){
            printf("b scored more\n");
}
 else {
            printf("they are equal\n");
}
           }
//end compare function
//sum function
int sum(int a1[]){
int a=0;
   for (int i = 0; i <= 2; i++){
             a=a+a1[i];
   }
   return a; //returning the sum to main
   }
//end sum function
```

2.5 Use files for the input data.
Solution:

```
//Code by GVV Sharma
//April 14 2025
//using files
#include <stdio.h>

//Declaring functions
void compare(int a,int b);
int sum(int a[]);

//begin main function
int main() {
          //Declaring arrays
int a1[3], b1[3];
//declare file pointer
FILE *fp;
int i;
//Initializing sums
int a=0,b=0;
```

```
            //Read a from file a.dat
            //Open file pointer
fp = fopen(”a.dat”, ”r”);

//load data from file to array a1
 for(i=0;i<=2;i++){
    fscanf(fp,”%d”,&a1[i]);
   }

//Cose file pointer

fclose(fp);
            //Read a from file b.dat
            //Open file pointer
fp = fopen(”b.dat”, ”r”);

//load data from file to array b1
 for(i=0;i<=2;i++){
    fscanf(fp,”%d”,&b1[i]);
   }
//Close file pointer
fclose(fp);

//finding sum for A
a = sum(a1);
//finding sum for B
b = sum(b1);
//Call compare function
   compare(a,b);
 return 0;
}
//end main function

//compare function
void compare(int a,int b){
              if (a>b){
              printf(”a␣scored␣more\n”);
}
 else if (a<b){
              printf(”b␣scored␣more\n”);
}
 else {
              printf(”they␣are␣equal\n”);
}
            }
```

```
//end compare function
//sum function
int sum(int a1[]){
int a=0;
   for (int i = 0; i <= 2; i++){
              a=a+a1[i];
   }
   return a; //returning the sum to main
   }
//end sum function
```

2.6 Revise the files program using pointer arrays
Solution:

```
//Code by GVV Sharma
//April 14 2025
//using pointer arrays
#include <stdio.h>
#include <stdlib.h>

//Declaring functions
void compare(int a,int b);
int sum(int a[], int m);

//begin main function
int main() {
//declare pointer arrays
int *a1,*b1,m = 3;
//Initializing sums
int a=0,b=0,i;
//File pointer
FILE *fp;

//Create a1
a1= (int *)malloc(m * sizeof( a1));
b1= (int *)malloc(m * sizeof( b1));

            //Read a from file a.dat
            //Open file pointer
fp = fopen("a.dat", "r");

//load data from file to array a1
 for(i=0;i<=2;i++){
    fscanf(fp,"%d",&a1[i]);
   }
```

```
//Cose file pointer

fclose(fp);
            //Read a from file b.dat
            //Open file pointer
fp = fopen("b.dat", "r");

//load data from file to array b1
 for(i=0;i<=2;i++){
    fscanf(fp,"%d",&b1[i]);
   }
//Close file pointer
fclose(fp);

//finding sum for A
a = sum(a1,m);
//finding sum for B
b = sum(b1,m);
//Call compare function
compare(a,b);

//free memory
free(a1);
free(b1);
 return 0;
}
//end main function

//compare function
void compare(int a,int b){
             if (a>b){
             printf("a scored more\n");
}
 else if (a<b){
             printf("b scored more\n");
}
 else {
             printf("they are equal\n");
}
            }
//end compare function
//sum function
int sum(int *vec,int m){
int a=0;
  for (int i = 0; i < m; i++){
```

```
                a=a+vec[i];
        }
        return a; //returning the sum to main
        }
//end sum function
```

2.7 Revise the files program using only functions

Solution:

```
//Code by GVV Sharma
//April 14 2025
//using functions for all
#include <stdio.h>
#include <stdlib.h>

//Declaring functions
void compare(int a,int b);
int sum(int a[], int m);
int *loadVec(char *str,int m);
int *createVec(int m);

//begin main function
int main() {
//Initializing sums
int a=0,b=0,m = 3;
//declare pointer arrays
int *a1,*b1;
        //Read a from file a.dat
a1= loadVec("a.dat",m);
b1= loadVec("b.dat",m);
        //Read b from file b.dat

//finding sum for A
a = sum(a1,m);
//finding sum for B
b = sum(b1,m);
//Call compare function
compare(a,b);
 return 0;
}
//end main function

//compare function
void compare(int a,int b){
        if (a>b){
        printf("a scored more\n");
}
 else if (a<b){
            printf("b␣scored␣more\n");
}
 else {
            printf("they␣are␣equal\n");
}
           }
//end compare function
//sum of vector elements
int sum(int ∗vec,int m){
int a=0;
  for (int i = 0; i < m; i++){
             a=a+vec[i];
  }
  return a; //returning the sum to main
  }
//end sum function
//loading file data into vector
int ∗loadVec(char ∗str,int m){
FILE ∗fp;
int i;
int ∗vec=createVec(m);
          //Open file pointer
fp = fopen(str, "r");

//load data from file to array a1
 for(i=0;i<m;i++){
   fscanf(fp,"%d",&vec[i]);
  }

//Cose file pointer

fclose(fp);
return vec;
}

//end loading file data into vector
//Defining the function for vector creation
int ∗createVec(int m)
{
 int ∗vec;

 //Allocate memory to the pointer
vec = (int ∗)malloc(m ∗ sizeof( vec));
```

```
  return vec;
}
```

Verify the following using ifelse

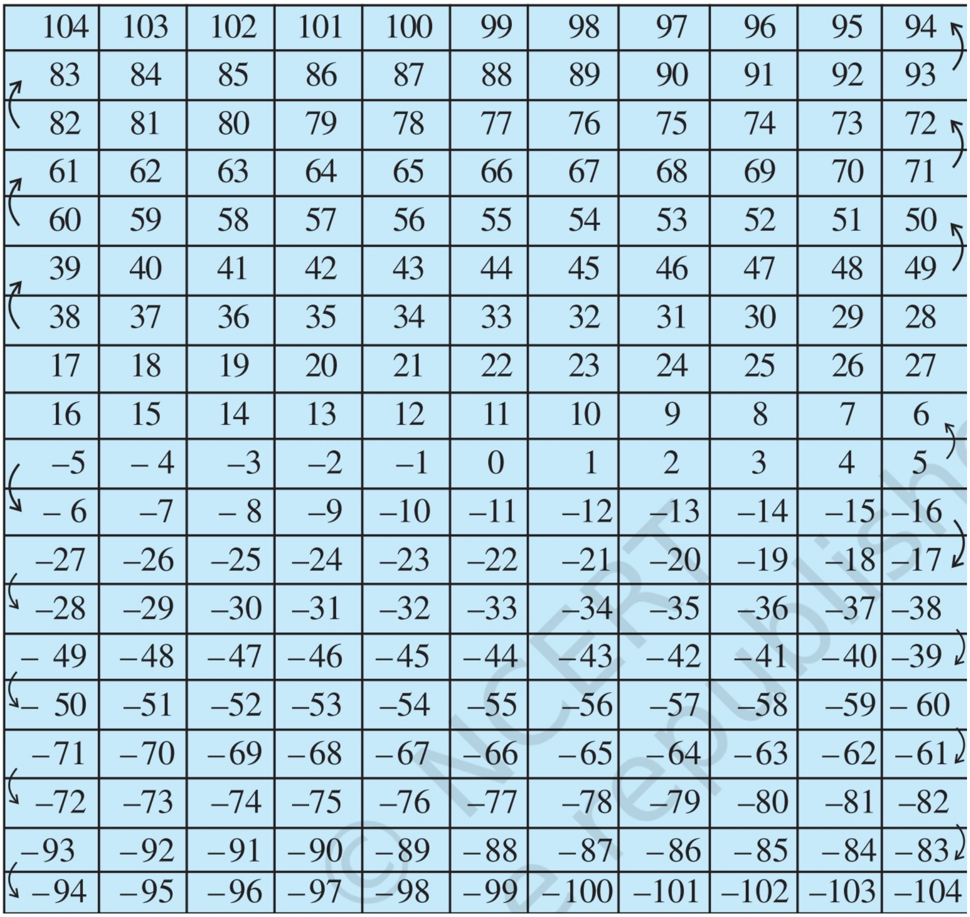
2.8 $25 \times (-21) = (-21) \times 25$
2.9 $(-23) \times 20 = 23 \times (-20)$
2.10 $(-15) \times [(-7) + (-1)] = (-15) \times (-7) + (-15) \times (-1)$
2.11 $10 \times [6 + (-2)] = 10 \times 6 + 10 \times (-2)$
2.12 $10 \times [6 - (-2)] = 10 \times 6 - 10 \times (-2)$
2.13 $(-15) \times [(-7) - (-1)] = (-15) \times (-7) - (-15) \times (-1)$
2.14 $18 \times [(7) + (-3)] = 18 \times (7) + 18 \times (-3)$
2.15 $(-21) \times [(-4) + (-6)] = (-21) \times (-4) + (-21) \times (-6)$

Use arrays for the following

2.16 $(-12) \times (-11)(10)$
2.17 $(9) \times (-3) \times (-6)$
2.18 $(-18) \times (-5) \times (-4)$
2.19 $(-1) \times (-2) \times (-3) \times (-4)$
2.20 $(-3) \times (-6) \times (-2) \times (-1)$

3 Random Numbers

3.0.21 Take a board marked from -104 to 104 as shown in the figure.
3.0.22 Take a bag containing two blue and two red dice. Number of dots on the blue dice indicate positive integers and number of dots on the red dice indicate negative integers.
3.0.23 Every player will place his/her counter at zero.
3.0.24 Each player will take out two dice at a time from the bag and throw them.
3.0.25 After every throw, the player has to multiply the numbers marked on the dice.
3.0.26 If the product is a positive integer then the player will move his counter towards 104; if the product is a negative integer then the player will move his counter towards -104.
3.0.27 The player who reaches either -104 or 104 first is the winner.



104	103	102	101	100	99	98	97	96	95	94
83	84	85	86	87	88	89	90	91	92	93
82	81	80	79	78	77	76	75	74	73	72
61	62	63	64	65	66	67	68	69	70	71
60	59	58	57	56	55	54	53	52	51	50
39	40	41	42	43	44	45	46	47	48	49
38	37	36	35	34	33	32	31	30	29	28
17	18	19	20	21	22	23	24	25	26	27
16	15	14	13	12	11	10	9	8	7	6
–5	– 4	–3	–2	–1	0	1	2	3	4	5
– 6	–7	– 8	–9	–10	–11	–12	–13	–14	–15	–16
–27	–26	–25	–24	–23	–22	–21	–20	–19	–18	–17
–28	–29	–30	–31	–32	–33	–34	–35	–36	–37	–38
– 49	–48	–47	–46	–45	–44	–43	–42	–41	–40	–39
– 50	–51	–52	–53	–54	–55	–56	–57	–58	–59	– 60
–71	–70	–69	–68	–67	–66	–65	–64	–63	–62	–61
–72	–73	–74	–75	–76	–77	–78	–79	–80	–81	–82
–93	–92	–91	–90	–89	–88	–87	–86	–85	–84	–83
–94	–95	–96	–97	–98	–99	–100	–101	–102	–103	–104

Fig. 3.0.1

3.0.28 Write a program to simulate the game. Give the inputs manually.

3.0.29 Revise the program by generating the inputs using randomly as follows

a) Generate the numbers on all the dice using a uniform distribution ranging from 1 to 6.

b) Simulate the blue and red dice through a Bernoulli distribution having values 1 and -1.