**Nguyễn Mạnh Bảo Tín MSSV SE170134**

**[tinnmbse170134@fpt.edu.vn](mailto:tinnmbse170134@fpt.edu.vn)**

**Subject: PRF192- PFC**

**Workshop 06**

**Objectives: Managing arrays**

**Sample: Canadian SIN (Social Insurance Number)**

SIN: **1**9**3** 4**5**6 **7**87 | check digit is **7** add first set of alternates to themselves

9 4 6 8 9 4 6 8 **18 8 12 16**

add the digits of each sum **1+8+8+1+2+1+6 = 27 (T1)**

add the other alternates 1+3+5+7 = 16 (T2)

total = T1+T2 = 27+16=43

Next highest integer multiple of 10 T3= 50 (50>43).

Difference T3-total = 50-43= **7** Matches the check digit, therefore this SIN is valid

SIN: 193456787

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N0 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 |  |  |  |
|  | 1 | **9** | 3 | **4** | 5 | **6** | **7** | **8** | **7** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **9** | **4** | **6** | **8** | **9** | **4** | **6** | **8** | **18** | **8** | **12** | **16** |
| C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 |

**Algorithm for checking whether a number is a Canadian SIN or not**

Use the array N, 10 elements, N[0] is not used

Use the array C, 12 elements, C[0] is not used

From n, computing N[i]:

From N, computing C[i]:

Compute

T1= C9/10 + C9%10 + C10/10 + C10%10 + C11/10 + C11%10 + C12/10 + C12%10

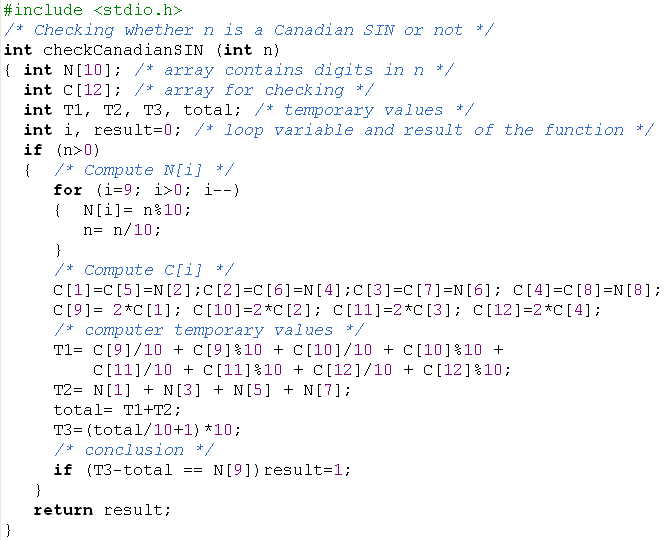
T2= N1 + N3 + N5 + N7;

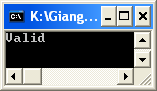
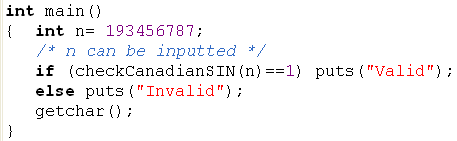
Total= T1+T2

T3= (Total/10+1) \*10; ( Total=**4**3 🡪 T3 = (**4**+1)\*10

If (T3-Total == N9) return “Valid”

else return “Invalid”





Refer to the sample above, write the following problem.

**Problem 1 (4 marks)**

An ISBN consists of exactly **10 digits**.  The rightmost digit is the check digit.  The check digit is validated modulo 11.

* multiply each digit from the first to the ninth by a weight from 10 to 2 respectively (the first digit by 10, the second by 9,..., the ninth by 2).
* the sum of the products plus the check digit should be divisible without remainder by 11.
* if there is a remainder, the whole number is not a valid ISBN.

Consider the following example:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ISBN 0003194876 | check digit is 6 add first set of alternates to themselves  0 0 0 3 1 9 4 8 7 10 9 8 7 6 5 4 3 2 **0 0 0 21 6 45 16 24 14** = 126  add check digit 6 total 132 divide by 11 12 remainder 0 Therefore this ISBN is valid. | | | | | | | | |
| I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 | **I10** |
| **0** | **0** | **0** | **3** | **1** | **9** | **4** | **8** | **7** | **6** |
|  |  |  |  |  |  |  |  |  |  |
| C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 |  |
| **0**\***10**=**0** | **0**\***9**=**0** | **0**\***8**=**0** | **3**\***7**=**21** | **1**\***6**=**6** | **9**\***5**=**45** | **4**\***4**=**16** | **8**\***3**=**24** | **7**\***2**=**14** |  |

T= C1+ C2+C3+C4+C5+C6+C7+C8+C9 + I10; (T=132)

If (T%11==0) print out “Valid” else print out “Invalid”

Write a program that will accept a number (>=1 000 000 000) then show whether the number is an ISBN or not.

ISBN Validator ============== ISBN (0 to quit): 0003194876

This is a valid ISBN.

ISBN (0 to quit): 0003194875

This is not a valid ISBN. ISBN (0 to quit): 0

Have a Nice Day!

**Program 1:**

#include <stdio.h>

#include <stdlib.h>

int checkISBN(int n)

{

int N[10];

int C[9];

int T = 0;

int i, j;

for (i = 9; i >= 0; i--)

{

N[i] = n % 10;

n = n / 10;

}

for (i = 0; i < 9; i++)

for (j = i; j < i+1; j++)

{

C[j] = N[i]\*(10-i);

T += C[j];

}

T += N[9];

if ((T % 11) == 0)

printf("Ma ISBN hop le.\n");

else

printf("Ma ISBN khong hop le.\n");

}

int main() {

int n, check;

check = 1;

printf("Phan mem kiem tra ma ISBN\n==============\n");

while (check == 1) {

do

{

printf("Nhap ma ISBN (nhap 0 de thoat phan mem): ");

scanf("%d", &n);

if (n < 0)

printf("Sai dinh dang, vui long nhap lai\n");

}

while (n < 0);

if (n != 0)

checkISBN(n);

else check = 0;

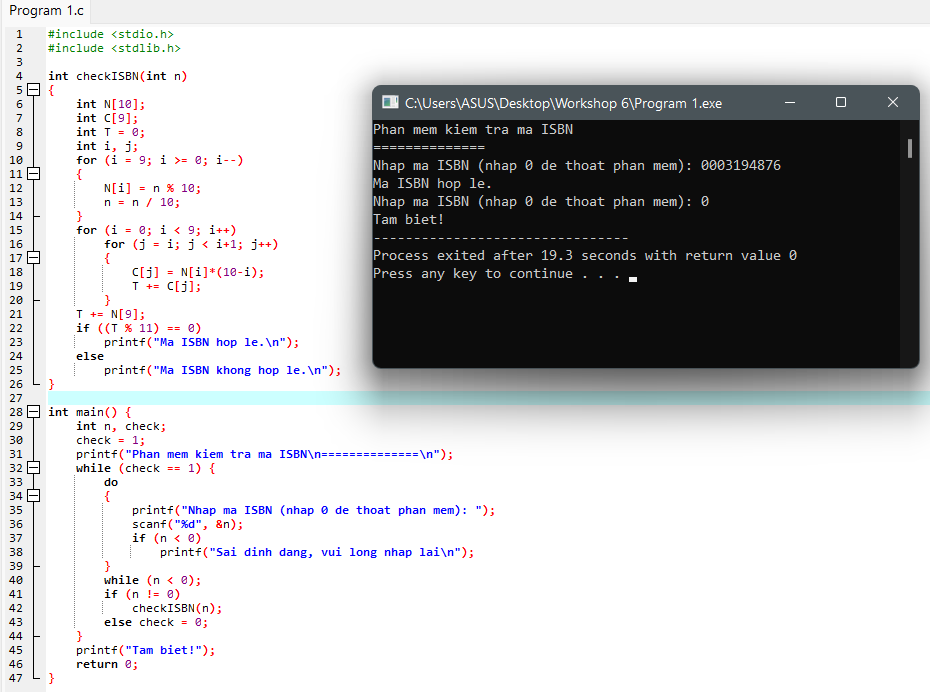
}

printf("Tam biet!");

return 0;

}

**Screenshots for Program 1:**



**Problem 2 (6 marks)**

Develop a C-program that helps user managing an 1-D array of real numbers(maximum of 100 elements) , with initial number of elements is 0, using the following simple menu:

1- Add a value

2- Search a value

3- Print out the array

4- Print out values in a range

5- Print out the array in ascending order

Others- Quit

* When the option 1 is selected, user will enters a value then it is added to the array
* When the option 2 is selected, user will enters a value then number of it’s existences will be printed out.
* When the option 3 is selected, values in the array will be printed out.
* When the option 4 is chosen, user will enter 2 values, minVal and maxVal, the values in array which are between minVal and maxVal are printed out (minVal <=value<=maxVal)
* When the option 5 is chosen, values in array will be printed out in ascending order but **their position are preserved. ( sorting based their pointers only)**

**Program 2:**

#include <stdio.h>

#include <stdlib.h>

#define MAXN 100

int menu()

{

int choice;

printf("\n=======================MENU======================");

printf("\n| 1- Add a value Press: 1 |");

printf("\n| 2- Search a value Press: 2 |");

printf("\n| 3- Print out the array Press: 3 |");

printf("\n| 4- Print out values in a range Press: 4 |");

printf("\n| 5- Print out the array in ascending Press: 5 |");

printf("\n| 6- Quit Press: 6 |");

printf("\n=================================================\n");

printf("\nEnter Your Choice: ");

scanf("%d", &choice);

fflush(stdin);

return choice;

}

int isFull(int n)

{

return n == MAXN;

}

int isEmpty(int n)

{

return n == 0;

}

void add(int values, int value[], int \*pn)

{

value[\*pn] = values;

(\*pn)++;

}

int search(int values, int value[], int n)

{

int i;

for (i = 0; i < n; i++)

if (value[i] == values)

return i;

return -1;

}

void printArray(int value[], int n)

{

printf("\nArray of number: \n");

int i, j;

for (i = 0; i < n; i++)

{

j = value[i];

printf("%d ", j);

}

}

void printRange(int value[], int n)

{

int maxVal, minVal, t = -1;

printf("\nEnter the range [a..b]: \n");

printf("\na = ");

scanf("%d", &minVal);

printf("\nb = ");

scanf("%d", &maxVal);

printf("\n");

int i;

for (i = 0; i < n; i++)

if (value[i] >= minVal && value[i] <= maxVal)

{

printf("%d ", value[i]);

t = i;

}

if (t < 0)

printf("No value between range!\n");

}

void printASC(int value[], int n)

{

int \*adds = (int\*)calloc(n, sizeof(int));

int i, j;

for (i = 0; i < n; i++)

adds[i] = value[i];

int t;

for (i = 0; i < n-1; i ++)

for ( j = n-1; j > i; j--)

if (adds[j] < adds[j-1])

{

t = adds[j];

adds[j] = adds[j-1];

adds[j-1] = t;

}

for (i = 0; i < n; i++)

printf("%d ", adds[i]);

free(adds);

}

int main()

{

int value[MAXN];

int n = 0;

int values;

int userChoice, check = 1;

do

{

userChoice = menu();

switch(userChoice)

{

case 1:

if (isFull(n))

printf("\nThe Array is full.\n");

else

{

printf("\nInput an added value: ");

scanf("%d", &values);

add(values, value, &n);

printf("\nAdded!\n");

}

break;

case 2:

if (isEmpty(n))

printf("\nThe Array is empty.\n");

else

{

int pos;

printf("\nInput the search value: ");

scanf("%d", &values);

pos = search(values, value, n);

if (pos < 0 || pos > n)

printf("\nNot found.\n");

else

printf("\nPosition is found: %d\n", pos);

}

break;

case 3:

if (isEmpty(n))

printf("\nThe Array is empty.\n");

else

printArray(value, n);

break;

case 4:

if (isEmpty(n))

printf("\nThe Array is empty.\n");

else

printRange(value, n);

break;

case 5:

if (isEmpty(n))

printf("\nThe Array is empty.\n");

else

printASC(value, n);

break;

default:

if (userChoice == 6)

check = 0;

else

printf("\nWrong input!\n");

}

}

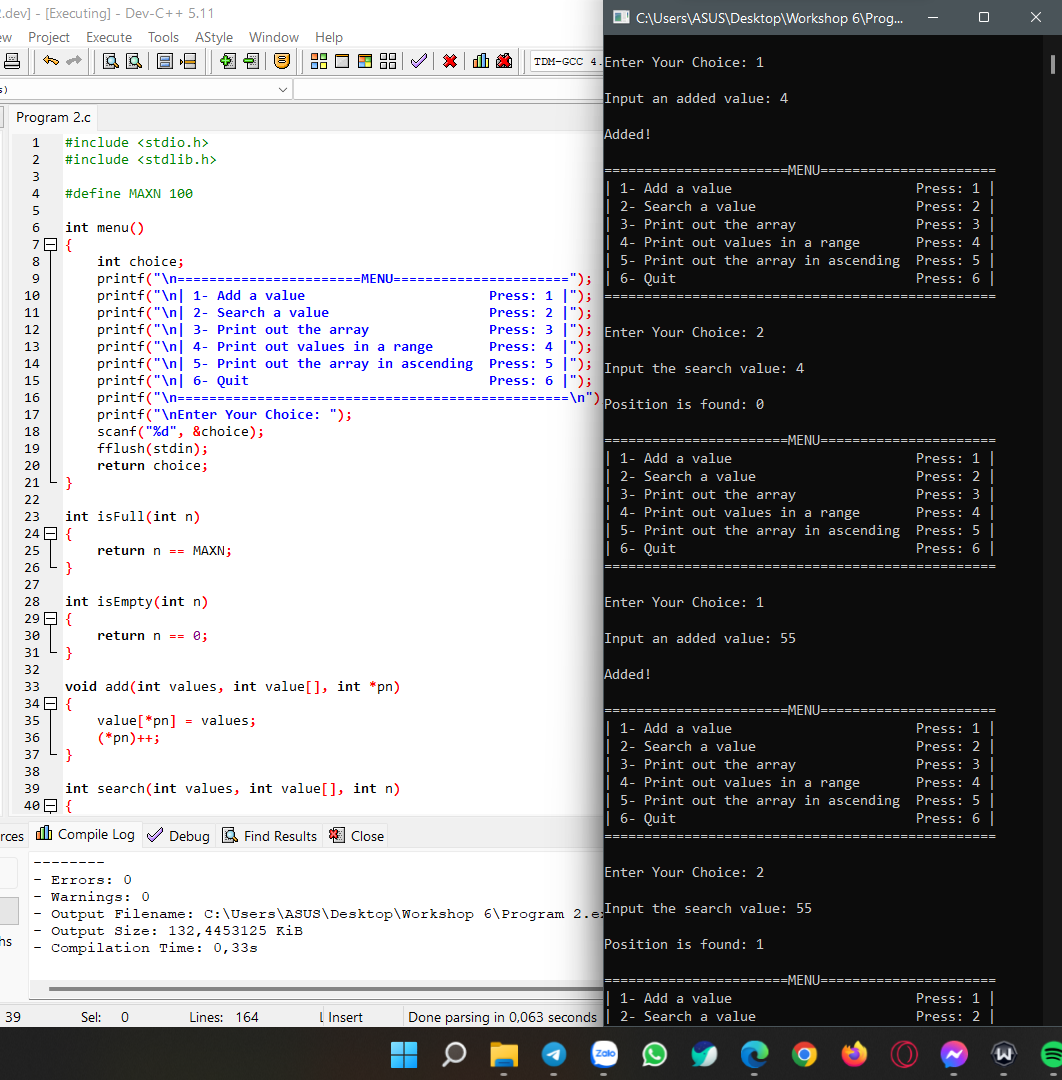
while (check == 1);

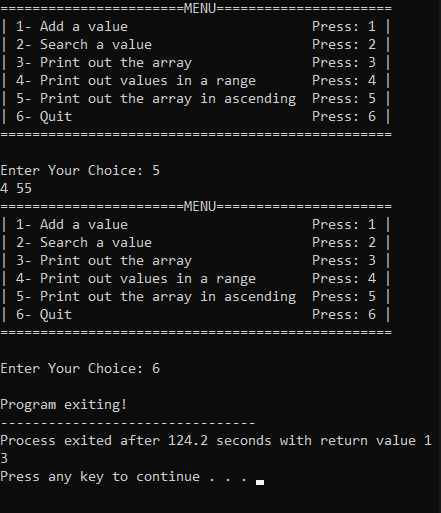
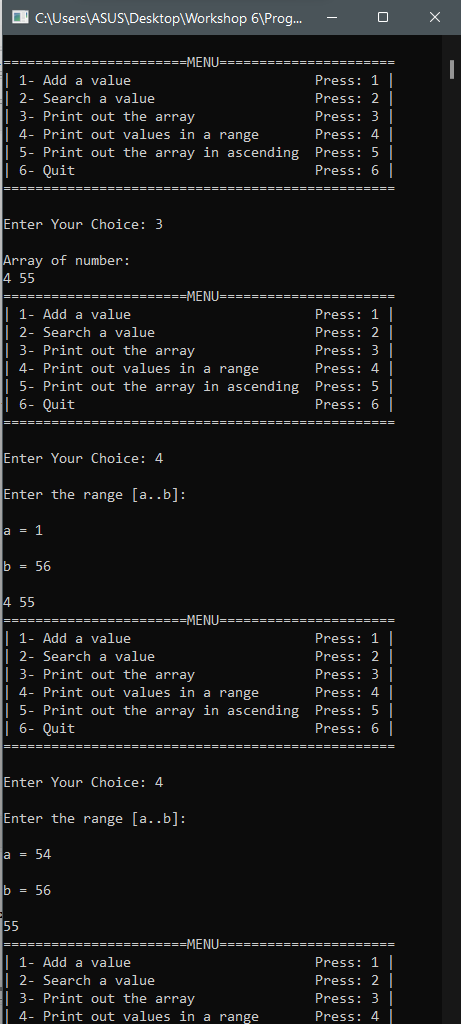
printf("\nProgram exiting!");

getch();

}

**Screenshots for Program 2:**





Related files can be found at: [Workshop 6 PRF192](https://fptuniversity-my.sharepoint.com/:f:/g/personal/tinnmbse170134_fpt_edu_vn/ElpeuW3oRwVGp0LhTQkp-IMBBseNMWxfCHmR9PN8Yhx4qQ?e=mDYjMi)