

Experiment 1: C programming language basics Data types, Variables, Arrays, Loops, Conditionals, Functions , Pointers , Structures

Objectives

The objectives of Experiment 1 are

- to learn C programming language basics Data types, Variables, Arrays, Loops, Conditionals, Functions , Pointers , Structures

Apparatus Required:

- Dev C++

Preliminary Work:

- Install required Dev C++ programme (<https://sourceforge.net/projects/orwelldevcpp/>)



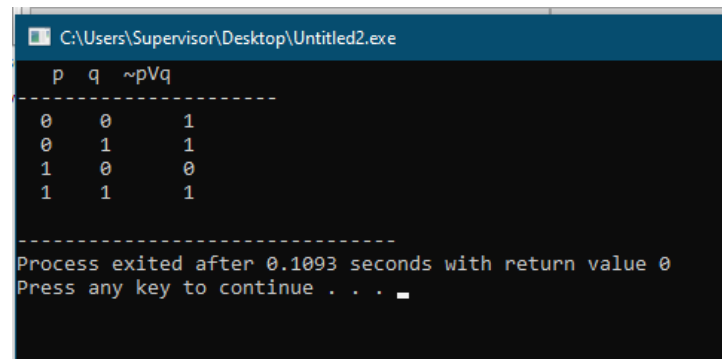
Figure 1

- Study the L03 notes.
- Write the following codes in the experimental work in Dev C++.
- Upload the codes to your flash memory and bring them to class.

Experimental Work:

1. Control Statements

Write the code that will calculate the truth table of the $\sim p \vee q$ proposition using control statements and display it on the screen.



```
C:\Users\Supervisor\Desktop\Untitled2.exe
p  q  ~pVq
-----
0   0   1
0   1   1
1   0   0
1   1   1

-----
Process exited after 0.1093 seconds with return value 0
Press any key to continue . . .
```

Figure 2: Program Output Window

Answer:

```
#include <stdio.h>

int main (void)
{
    int true=1, false=0;
    printf(" p q ~pVq\n");
    printf("-----\n");
    printf("%3d%5d%7d \n", false, false, !false || false);
```

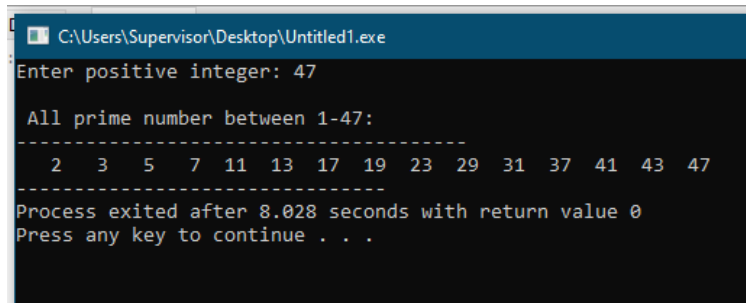
```

printf("%3d%5d%7d \n", false, true, !false || true);
printf("%3d%5d%7d \n", true, false, !true || false);
printf("%3d%5d%7d \n", true, true, !true || true);
return (0);
}

```

2. Loops (for-while) & Conditionals

- a. Write the C program that takes the n value, which is a positive integer, as input and find all prime numbers up to n and display it on the screen.



```

C:\Users\Supervisor\Desktop\Untitled1.exe
Enter positive integer: 47

All prime number between 1-47:
-----
 2  3  5  7 11 13 17 19 23 29 31 37 41 43 47
-----
Process exited after 8.028 seconds with return value 0
Press any key to continue . . .

```

Figure 3: Program Output Window

Answer:

```

#include <stdio.h>

int main (void)
{
    int n;

    int num, p,i;

    printf("Enter positive integer: ");

```

```

scanf("%d", &n); /*Girdinin alınmasi*/

printf("\n All prime number between 1-%d: ",n);

printf("\n-----\n");

for( num = 2; num <= n; num = num+1)

/*Let's assume that the number is prime: p=1 means the number is prime, p=0 means it is not prime*/

p=1; /*Assume the number is prime*/

i=2; /*The variable i is used to control which numbers the number entered by the user can be divided by*/

while ((i<num) && p==1)

{

    if (num%i == 0)

        p=0; /*The number is not prime because it is divisible.*/

        i=i+1;

}

if (p==1)

printf("%4d" , num);    }

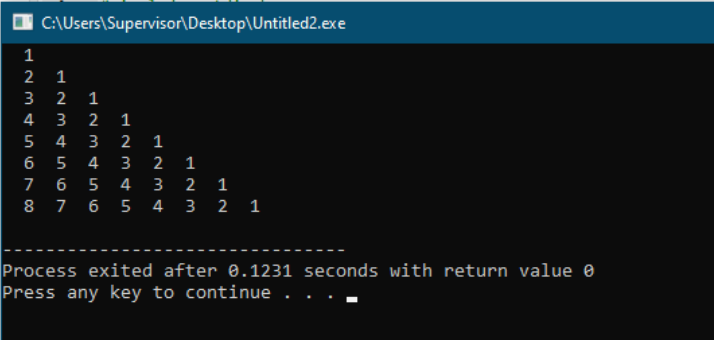
return (0);

}

```

b. Let's write the code that outputs the following using nested loops.

```
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
```



```
C:\Users\Supervisor\Desktop\Untitled2.exe
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
-----
Process exited after 0.1231 seconds with return value 0
Press any key to continue . . .
```

Figure 4: Program Output Window

Answer:

```
#include <stdio.h>
int main (void)

{
    int i,j;
    for (i=1; i<=8; i++)
    {
        for (j=i; j>=1; j--)
            printf("%3d", j);
        printf("\n");
    }
}
```

```
        return(0);  
    }
```

3. Functions

Write a code that takes a positive integer value from the user and shows the number of digits of the integer on the screen. While writing the code, use the function named ***basamak_bul ()***. *basamak_bul ()* must take an integer value from where it was called, find the number of digits of the integer and return it to where it was called.

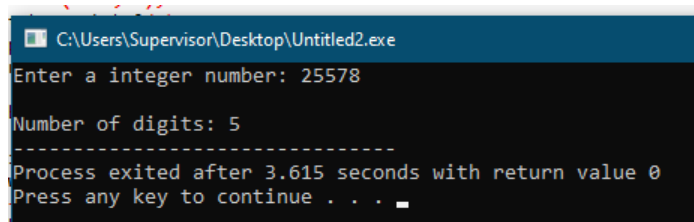


Figure 5: Program Output Window

Answer:

```
#include <stdio.h>  
  
int basamak_bul(int x);  
  
int main (void)  
{  
  
    int a,t;  
  
    printf("Enter a integer number: ");  
  
    scanf("%d",&a);
```

```

        t=basamak_bul(a);

        printf("\nNumber of digits: %d",t);

        return(0);
    }

int basamak_bul(int x)
{
    int b,toplam = 0;

    while (x)
    {
        b=x%10;

        toplam = toplam+1;

        x=x/10;

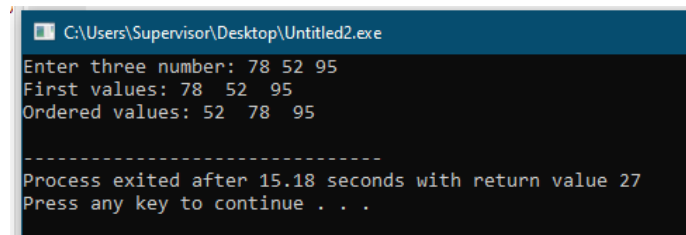
    }

    return(toplam);
}

```

4. Pointers

- a. Write the C program that orders 3 integers received from the user and displays the first input values and sequential values on the screen. Send back the sequential numbers as source parameters. (Use the **pointers** for this.) Use the **replace ()** function to sort the 3 numbers.



```
C:\Users\Supervisor\Desktop\Untitled2.exe
Enter three number: 78 52 95
First values: 78 52 95
Ordered values: 52 78 95

-----
Process exited after 15.18 seconds with return value 27
Press any key to continue . . .
```

Figure 6: Program Output Window

Answer:

```
#include <stdio.h>
/* Program that sorts the 3 entered numbers*/
void replace(int*,int*);
int main (void)
{
    int x,y,z;
    printf("Enter three number: ");
    scanf("%d%d%d", &x,&y,&z);
    printf("First values: %d %d %d\n", x,y,z);
    if (x>y) replace(&x,&y);
    if (x>z) replace(&x,&z);
    if (y>z) replace(&y,&z);
    printf("Ordered values: %d %d %d\n", x,y,z);
}

/*replace() function that changes the values of two parameters*/
void replace(int *a, int *b)
/*a and b are taken as source parameters. Therefore, changes in this function will be reflected in the sent parameter.*/
{
    int temporary;
    temporary = *a;
```

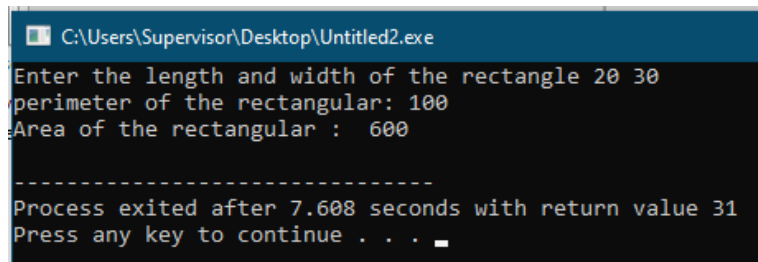


```

    *a = *b;
    *b = temporary;
}

```

- b. Write a function that finds the perimeter and area of a rectangle. In the function, take the width and length of the rectangle as the value parameter and return the perimeter and area as the source parameter.



```

C:\Users\Supervisor\Desktop\Untitled2.exe
Enter the length and width of the rectangle 20 30
perimeter of the rectangular: 100
Area of the rectangular : 600

-----
Process exited after 7.608 seconds with return value 31
Press any key to continue . . . 

```

Figure 7: Program Output Window

Answer:

/* Program to find the perimeter and area of a rectangle. While the width and height of the rectangle are taken as value parameters, the perimeter and area are returned as source parameters */

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

```
void perimeter_area(int, int , int*, int*);
```

```
int main (void)
```

```
{
```

```
    int width,length,perimeter,area;
```

```
    printf("Enter the length and width of the rectangle ");
```

```
    scanf("%d%d",&length,&width);
```

```

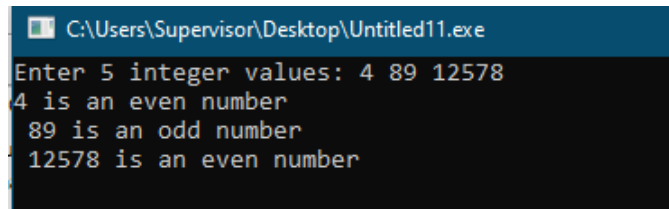
if (length<0 || width<0)/*girdi kontrolü*/
printf("\nYou entered the wrong value");
else
{
    perimeter_area(width, length , &perimeter, &area);
    printf("perimeter of the rectangular: %d\n", perimeter);
    printf("Area of the rectangular : %d\n", area);
}
}

void perimeter_area(int w, int l, int *e, int *a)
{
    *e = 2* (w+l); /*Calculate perimeter*/
    *a = w*l; /*Calculate area*/
}

```

5. Arrays

Write a C program in which the user enters integers in a 5-element array and after each integer value entered in the array, it shows whether the entered number is odd or even. This program should consist of two functions. The **find ()** function must take an integer value from where it is called and show it is odd or even. The **main ()** function should take 5 integer values from the user, store them in an array and using the **find()** function, show that the array elements are odd or even.



```
C:\Users\Supervisor\Desktop\Untitled11.exe
Enter 5 integer values: 4 89 12578
4 is an even number
89 is an odd number
12578 is an even number
```

Figure 8: Program Output Window

Answer:

```
#include<stdio.h>

void bul (int);

int main (void)
{
    int k[5],i;

    printf("Enter 5 integer values: ");

    for (i = 0; i < 5; i++)
    {
        scanf("%d",&k[i]);

        bul(k[i]);
    }

    return(0);
}
```

```

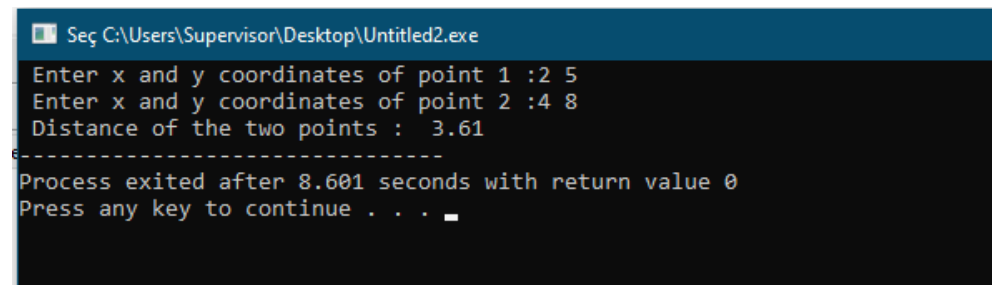
void bul (int a)
{
    if (a%2 == 0)
        printf("%d is an even number/n ", a);
    else
        printf("%d is an odd number/n ", a);
}

```

6. Structures

Write a program that takes the coordinates of two points (x1,y1) and (x2,y2) as input from the user and calculates the distance between them. The x and y coordinates of each point in your program should be kept in a struct. The distance formula is as follows:

$$\text{distance} = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$



```

Seq C:\Users\Supervisor\Desktop\Untitled2.exe
Enter x and y coordinates of point 1 :2 5
Enter x and y coordinates of point 2 :4 8
Distance of the two points : 3.61
-----
Process exited after 8.601 seconds with return value 0
Press any key to continue . . .

```

Figure 9: Program Output Window

Answer:

```
#include <stdio.h>

#include <math.h>

int main (void)
{
    struct point
    {
        int x,y;
    };
    struct point p1, p2;
    float distance;

    /*Read the coordinates of two points*/
    printf(" Enter x and y coordinates of point 1 :");
    scanf ("%d %d", &p1.x, &p1.y);
    printf(" Enter x and y coordinates of point 2 :");
    scanf ("%d %d", &p2.x, &p2.y);

    /* Calculate the distance between two points*/
    distance = sqrt (((p1.x-p2.x)*(p1.x-p2.x))+((p1.y-p2.y)*(p1.y-p2.y)));
    printf(" Distance of the two points : %5.2f", distance);
```

```
return 0;
```

```
}
```

IMPORTANT NOTE:

There will be a quiz about the experiment at the end of the lesson. You will be expected to write the requested code within the given time. Therefore, it is important that you understand and apply the relevant program codes in the part of experimental work .