Programming

Switch and Loops

Example – Fibonacci series

- Write a program that prints Fibonacci series elements which are smaller than 5
- Fibonacci series:
  - Fib(1) = 0
  - Fib(2) = 1
  - Fib (n) = Fib(n-1) + Fib(n-2)

Example – Fibonacci series

0 1 1 2 3 5 8 13 21 34 …

0+1+1 2 3 5 8 13+21 34

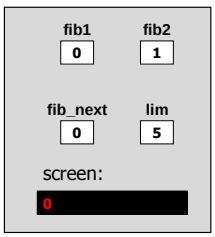
סלט פיבונאצ'י

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

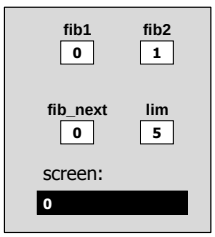


Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```



Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 0 fib2: 1
fib_next: 0 lim: 5
screen: 0 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 0 fib2: 1
fib_next: 1 lim: 5
screen: 0 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 fib2: 1
fib_next: 1 lim: 5
screen: 0 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 fib2: 1
fib_next: 1 lim: 5
screen: 0 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 fib2: 1
fib_next: 1 lim: 5
screen: 0 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 fib2: 1
fib_next: 1 lim: 5
screen: 0 1 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 | fib2: 1

fib_next: 2 | lim: 5

screen:

0 1 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 | fib2: 1

fib_next: 2 | lim: 5

screen:

0 1 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 | fib2: 2

fib_next: 2 | lim: 5

screen:

0 1 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 | fib2: 2

fib_next: 2 | lim: 5

screen:

0 1 1

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 | fib2: 2

fib_next: 2 | lim: 5

screen:

0 1 1 2

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 1 | fib2: 2

fib_next: 3 | lim: 5

screen:

0 1 1 2

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 2 fib2: 2
fib_next: 3 lim: 5
screen:
0 1 1 2

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 2 fib2: 3
fib_next: 3 lim: 5
screen:
0 1 1 2

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 2 fib2: 3
fib_next: 3 lim: 5
screen:
0 1 1 2

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 2 fib2: 3
fib_next: 3 lim: 5
screen:
0 1 1 2 3

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

fib1: 2 fib2: 3
fib_next: 5 lim: 5
screen:
0 1 1 2 3

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

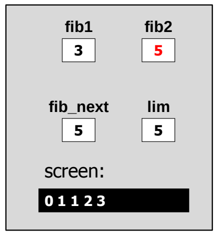
fib1: 3 fib2: 3
fib_next: 5 lim: 5
screen:
0 1 1 2 3

Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

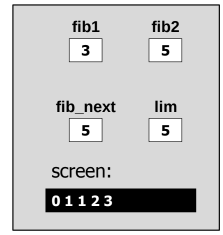


Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```

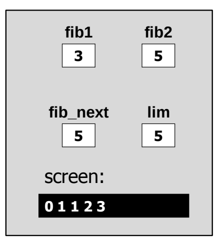


Fibonacci – step by step

```
int fib1 = 0, fib2 = 1,
    fib_next = 0,
    lim = 5;
...
printf("%d ", fib1);

while (fib2 < lim)
{
    printf("%d ", fib2);
    fib_next = fib1 + fib2;
    fib1 = fib2;
    fib2 = fib_next;
}

printf("\n");
```



Exercise : print a series

- Write a program that prints the first 10 elements in the following series :

  1 -2 3 -4 5 -6 7 -8 9 -10……

Solution (using while)

`printSeries_whileExample.c`

Solution (using for)

`printSeries_forExample.c`

Example 1: Increasing Series

- Write a program that gets positive integers from the user and prints an increasing series (without duplications):
- Every number is compared to its previous as follows:
- Number > previous ➔ print it
- Number == previous ➔ ignore it
- Number < previous ➔ stop

Solution 1: Increasing Series

```
#include <stdio.h>
int main() {
   int x, prev = -1;
   printf ("please enter a series of positive integers :\n");
   do {
        scanf("%d", &x );
        if (x < prev)
              break;
        if (x == prev)
              continue;
        printf("%d ", x);
        prev = x;
   } while (x  >= prev);
   return 0;
}
```

Example 2: Prime Numbers

- Write a program that prints all the prime numbers smaller than or equal to N.
- N is given by the user (input)
  - How do we go over all the numbers <= N?
  - How do we know if a number is prime?

Solution 2 : Prime Numbers

```
int main()
{
   int candidate = 0, divisor = 0, last = 0;

   printf("Enter a number\n");
   scanf("%d", &last);

   for (candidate = 2; candidate <= last; ++candidate)
   {
        for (divisor = 2; divisor < candidate; ++divisor)
        {
           if (candidate % divisor == 0)
              break;
        }
        if (divisor == candidate)
            printf("the number %d is prime\n", candidate);
   }
   return 0;
}
```

Example 3 : A Calculator

- Write a program that receives two numbers from the user and an operator (as character)
- The program checks that none of the numbers is zero
- The program display the expression and its result

Solution 3: A calculator

```
int main( )
{
     double n1=0.0, n2=0.0, res=0.0;
     char op=0;

     do {
           printf("Please enter two numbers: ");
           scanf("%lf%lf", &n1, &n2);
     }while(n1 == 0 || n2 == 0);

     printf("Please enter an arithmetical
              operator (+, -, * or /): ");
     scanf(" %c", &op);
```

Solution 3 Cont.

```
switch(op) {
   case '+':
     res = n1 + n2;
     break;
   case '-':
     res = n1 - n2;
     break;
   case '*':
     res = n1 * n2;
     break;
   case '/':
     res = n1 / n2;
     break;
    default:
       printf("%c is an
       invalid arithmetical
       operator!\n", op);
       return 1;
  }

  /*  Display the
  expression and its result
  */
  printf("%lf %c %lf =
  %lf\n", n1, op, n2, res);

  return 0;
}
```

getchar

- **getchar()** gets a single character from the user.
- Requires including **stdio.h**
- Returns a non-positive number on failure.
- Similar to **scanf**.

```
char c;
c = getchar();
```

<=====>

```
char c;
scanf("%c", &c);
```

putchar

- **putchar(char c)** prints out the character inside the brackets.
- Requires including **stdio.h**
- Similar to **printf**.

```
char c;
putchar(c);
```

<=====>

```
char c;
printf("%c", c);
```

Example 4: Copy

```
int main()
{
    int c = 0;

    c = getchar();
    while (c != '\n')
    {
        putchar(c);
        c = getchar();
    }

    return 0;
}
```

Example 5: Counting Letters

- Write a program that read input from the user until end-of-line ('\n') is encountered.
- The program counts the number of letters before the first space (or until the end-of-line, if space is not found).

Example 5: Counting Letters

```
int main()
{
    int counter = 0, c = 0;

    printf("Enter a line of text:\n");

    c = getchar();
    while (c != '\n')
    {
          /* found the first space - exit the loop */
          if (c == ' ')
             break;
          ++counter;
          c = getchar();
    }

    /* we found a space */
    if (c == ' ')
          printf("There are %d letters before the first space.\n", counter);
    /* Didn't find a space */
    else
          printf("%d letters before end-of-line(no spaces in the input line)\n");

    return 0;
}
```

ctype library

- The <ctype.h> library contains character classification functions.
- Useful functions:
  - **`int isalpha (int c)`**
    test for alphabetic character
  - **`int isdigit(int c)`**
    test for digit
  - **`int tolower(int c)`**
    convert character to lowercase
  - **`int toupper(int c)`**
    convert character to uppercase
  - and more…
    see ctype.h library

Exercise 6: Low to Up

- Read input from the user until end-of-line (‘\n’) is encountered
- Convert lowercase letters to uppercase

Solution 6: Low to Up

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

int main(){
   int c = 0;
   printf ("Please enter a string: ");
    c = getchar();

   while (c != '\n'){
        putchar((c >= 'a'  &&  c <= 'z') ? toupper(c) : c);
        c = getchar();
   }
   putchar('\n');

   return 0;
}
```