

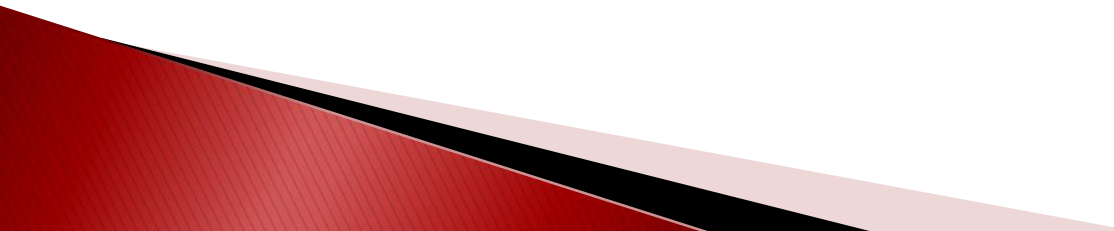
# Introduction to Informatics

Piroska Biró

# EXAMS

- ▶ End-term Test
  - 10th December 2014 – F01 – 8:00
- ▶ Retake Midterm Test
  - 19th December 2014 – F01 – 8:00
- ▶ Retake End-term Test
  - 19th December 2014 – F01 – 10:00
- ▶ Final exams
  - 8th January 2015 – F0 – 8:00
  - 26th January 2015 – F0 – 14:00
  - 30th January 2015 – F01 – 8:00

# Arrays

- ▶ a data structure, which provides the facility to store a collection of data of same type under single variable name
  - ▶ the size should be an individual constant
  - ▶ the index specifies the location of the element in the array
  - ▶ the array index starts from zero
  - ▶ the maximum index value will be equal to the size of the array minus one
- 

# One dimensional array

- ▶ declaration form of one-dimensional array is:

**data\_type array\_name[size];**

- ▶ `int numbers[5];`  
`numbers[0] = 1;      // set first element`

# One dimensional array

Example:

```
int a[10];  
float b[20];  
char c[100];
```

► initialization:

```
int a[10]={1,2,3,4,5,6,7,8,9,10};  
float f[100]={3.14, -12, 45};  
int b[50]={0};
```

# One dimensional array

- ▶ Input/Output one dimensional array

```
int i, n, a[100];  
    printf("n=");  
    scanf("%d", &n);  
for (i=0; i<n; i++)  
{  
    printf("Az %d. element:", i);  
    scanf("%d", &a[i]);  
}  
for (i=0; i<n; i++)  
    printf(" %d  ", a[i]);
```

# Exercise

- ▶ In the case of two given one dimensional arrays determine the sum of the arrays in a third array. And copy the values to another array and print the result.

# Solution

```
int a[8] = { 12, 24, 11, 7, 4, 13, 18, 52 };  
int b[8] = { 2, 44, 21, 17, 24, 3, 38, 11 };  
int c[8], d[8], i;
```

```
for (i = 0; i < 8; i++)  
    c[i] = a[i] + b[i];
```

```
for (i = 0; i < 8; i++)  
    d[i] = c[i];
```

```
for (i = 0; i < 8; i++)  
    printf("%d, ", d[i]);
```



# Exercise

What is the result of this code?

```
int B[40], i=0;  
for ( ; i < 30; ++i )  
    B[i] = 2*i+2;  
printf("%d\n%d\n", B[2], B[i-1 2]);
```

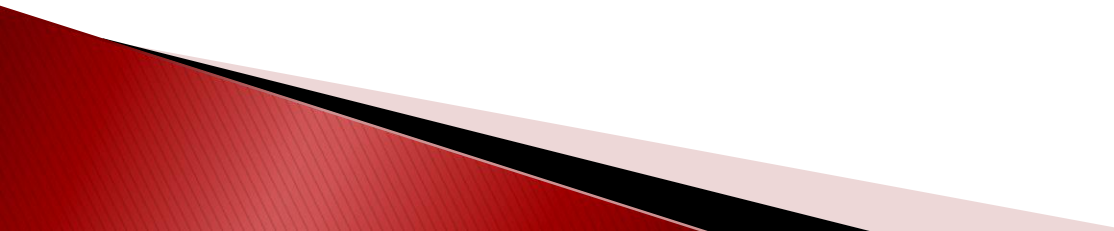
# Exercise

What is the result of this code?

```
int B[25], i=0;  
for ( ; i < 10; ++i )  
    B[i] = 4*(2*i+1);  
printf("%d\n%d\n", B[11], B[i-7]);
```

# Functions

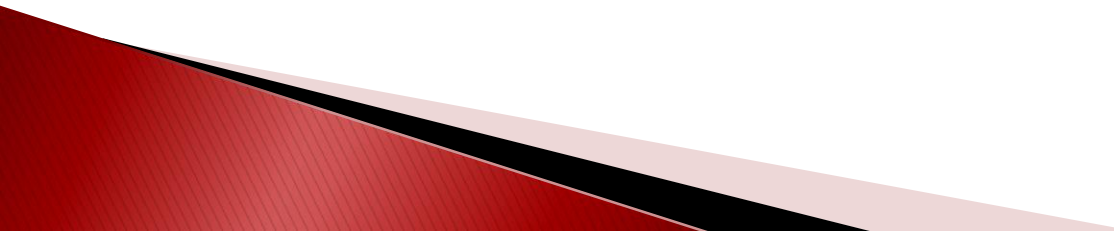
```
return_type function_name (type1 arg1, type2  
    arg2 ,..., typen argn)  
{  
    local variables;  
    statements;  
    return expression;  
}
```



# Functions

```
int add(int x, int y)
{
    int z;
    z=x+y;
    return z;    //return x+y;
}
```

```
double average(int x, int y, int z)
{
    double avg;
    avg=(x+y+z)/3;
    return avg;
}
```



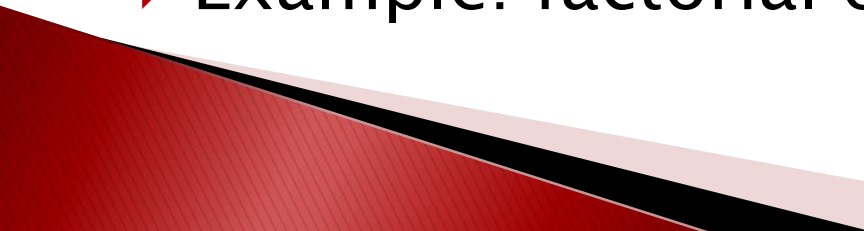
# Example

```
▶ int add(int x, int y)      //This function returns sum of a and b.
{
    return(x+y);
}

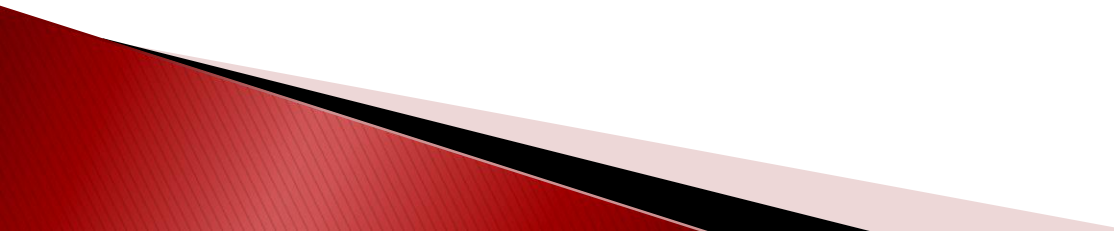
void display(int x, int y, int z). //This function returns nothing.
{
    printf("Sum of %d and %d is %d",x,y,z);
}

▶ int main()
{
    int a,b,c;
    printf("Enter two numbers\n");
    scanf("%d%d",&a,&b);
    c=add(a,b);
    display(a,b,c);
    return 0;
}
```

# Recursive functions

- ▶ call itself within that function
  - ▶ recursive function must have the following type of statements
    - a statement to test and determine whether the function is calling itself again.
    - a statement that calls the function itself and must be argument.
    - a conditional statement (if-else)
    - a **return** statement.
  - ▶ Example: factorial of a number.
- 

# Exercise

- ▶ Write a function which calculates the  $n$  factorial.
  - ▶ Write a function which returns the  $n^{\text{th}}$  fibonacci numbers.
  - ▶ Solve both tasks iteratively and recursively.
- 

# Functions

**`/*iterative*/`**

```
int factorial(int n)  
{ int f=1,i;  
    if (n==0 || n==1)  
        return 1;  
    else  
    {  
        for (i=1; i<=n; i++)  
            f=f*i;  
        return f;  
    }  
}
```

**`/*recursive*/`**

```
int factorial(int n)  
{ int f=1,i;  
    if (n==0 || n==1)  
        return 1;  
    else  
        f=n*factorial(n-1);  
    return f;  
}
```



# Recursion

**`/*iterative*/`**

```
int fibonacci (int n)  
{  
    int i, f1=0, f2=1, f3;  
    for (i=3;i<=n;i++)  
        {f3=f1+f2;  
        f1=f2;  
        f2=f3;}  
    return f3;  
    }
```

**`/*recursive*/`**

```
int fibonacci (int n)  
{  
    if (n==0 || n==1)  
        return 1;  
    else  
        return fibonacci(n-1)  
        +fibonacci(n-2);  
    }
```

# Pointers

- ▶ address of the variable

## Definition:

```
type *pointer;
```

## Example:

- `int i, *pi;`
- `float f, *pf;`
- `long double *pld;`

# Reference

With **&** operator we can refer to the adress of the variable.

Using the **\*** operator we can refer to the variable value.

**Sintax:**

- **&** variable

**Example:**

- `int a , *p;`
- `p = &a;`

We say that the **p** pointer points at the **a** variable.

# Pointers and arrays

- ▶ `int a[10];`
- ▶ `int *pa;`
  
- ▶ `pa=&a[0]` – 0. element address;
- ▶ `pa+1=&a[1]` – 1. element address;
- ▶ `pa+i=&a[i]` – i. element address;
- ▶ `*pa=a[0]` – 0. element;
- ▶ `*(pa+1)=a[1]` – 1. element
- ▶ `*(pa+i)=a[i]` – i. element
- ▶ `pa++;` – point to the next element
- ▶ `*(pa+i)++;` – increases the value of i. element with 1

# Exercise

- ▶ How much is the value of a variable after the evaluation of the code.

```
int a = 5;  
int *p;  
p=&a;  
*p=a+*p-2;  
printf("%d\n",a);
```

# Exercise

- ▶ How much is the value of x variable after the evaluation of the code.

```
int x = 7;  
int *p, **q;  
p=&x;  
q=&p;  
x -= *p + **q;  
printf("%d\n",x);
```

# Two dimensional arrays

- ▶ Table, matrix: rows, columns
  - every element has an index, so that we can refer to them
- ▶ Definition:
  - **type** name[row size][column size];
- ▶ Example:

```
int a[10] [10];  
float b[20] [20];  
char c[30] [30];
```
- ▶ Initialization:

```
int a[2][3]={1,3,5,0,77,-12};  
float b[50][50]={0};
```
- ▶ Reference:

```
name [row_index] [column_index];
```

 Example: `a[2][3];`

# Two dimensional array

**Example:** Print the following two dimensional array with two decimal precision!

```
float a[ 3 ][ 4 ] = {{ 2.5, 0.0, 5.5, 4.9 },  
                     { -4.2, 1.0, 0.2, 2.6 },  
                     { 2.7, -1.0, -5.1, 0.4 } };
```

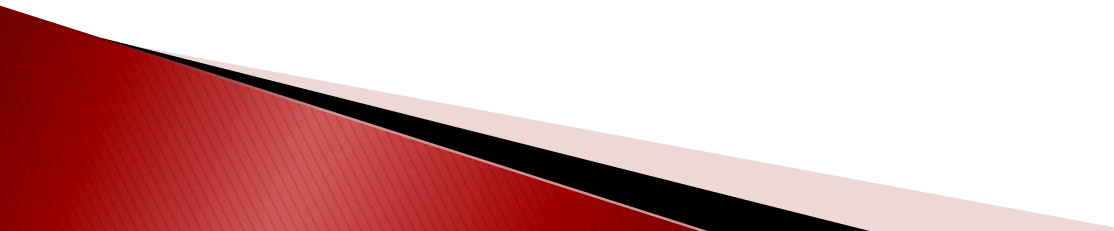
***Help:*** the declaration of the array can be given according to the example above, use the **float** type print **"%.2f "** with two decimal precision!



# Solution

```
int i, j;
float a[ 3 ][ 4 ] = { { 2.5, 0.0, 5.5, 4.9 },
                      { -4.2, 1.0, 0.2, 2.6 }, { 2.7, -1.0, -5.1, 0.4 } };

for(i=0; i<3; i++)
{
    for(j=0; j<4; j++)
        printf("%.2f ",a[i][j]);
    printf("\n");
}
```



# Exercise

What is the result after the execution of this code?

```
double m[4][3]={9, 7, 0.5, 3.14, 2, 5.0,  
4, 6.5, 0, 8, 5.95, 7.2};  
printf("%.2f , %.4f \n", m[1][1], m[3][1]);
```

# Exercise

What is the result after the execution of this code?

```
double m [3][5]={5.5, 7.9, 5, 7, 34, 10,  
11, -5, 0, 6.2, 0,15, 5, 5, 55.5};  
printf("%.3f , %f \n", m[2][1], m[1][3] );
```

# Exercise

What is the result after the execution of this code?

```
double m [4][3]={9.5, 7, 5, 6, 4.3, 7.4,5.5,2,1,  
                0.5,3};  
printf("%.2f , %f \n", m[2][1], m[1][2] );
```

# Symbolic constant

- ▶ name that substitute for a sequence of character that cannot be changed

## Example

```
#define PI 3.141593  
#define TRUE 1  
#define FALSE 0
```

# Macros

## Macro Declaration:

`#define name text`

## Example

`#define abs(x) ((x) < 0? (-(x)): (x))`

`#define min(a, b) ((a) < (b)? (a): (b))`

`#define square(x) ((x) * (x))`



# Exercise

What is the result after the execution of this code?

```
int B[60], i=0;  
for ( ; i < 30; ++i )  
    B[i] = (2*i+2)/2;  
printf("%d\n%d\n", *(B+5), B[i-4]);
```

# Exercise

What is the result after the execution of this code?

```
int A[45], i=0;  
for ( ; i < 45; ++i )  
    A[i] = 2*(i-3);  
printf("%d\n%d\n", *(A+25), A[i-3]);
```



# Exercise

What is the result after the execution of this code?

```
int B[25], i=0;  
for ( ; i < 20; ++i )  
    B[i] = 4*(2*i+1);  
printf("%d\n%d\n", *(B+11), B[i-7]);
```

# Exercise

What is the result after the execution of this code?

```
int i=-5, j=0, k=10;  
    if (k||i&&j)  
        { i++;  
          j+=k; }  
    else  
        k=k*(i+j);  
    printf("i=%d\tj=%d\tk=%d\n",i,j,k);
```

# Exercise

What is the result after the execution of this code?

```
int k=1, j=0, i = 0;
while (i<=122)
    if (i=1)
        j=k++;
    else
        i++;

printf("j=%d\ti=%d\n",j,i);
```

# Exercise

What is the result after the execution of this code? Explain your answer, please!

```
int i=5;  
    while (i= =6) i=6;  
    printf("%d\n",i);
```

# Exercise

What is the result after the execution of this code? Explain your answer, please!

```
int i=5;  
    while (i==6) i=6;  
    printf("%d\n",i);
```

# Exercise

What is the result after the execution of this code? Explain your answer, please!

```
int i;  
for (i=1; i!=10; i--)  
printf ("%d\n",i);
```

# Exercise

What is the result after the execution of this code? Work with the newly calculated values.

```
int a, b, c;  
a = b = c = 9;  
b = ++b + (a/5);  
c = c < a ? a+2 : b%5;  
b+=a; a%=2; --c;
```

# Exercise

What is the result after the execution of this code?

```
double a=5.12, b=42.23;  
    a-=b;  
    b+=a;  
    a=b-a;  
printf("\na= %.4lf \nb=%.2f",a,b);
```



# Exercise

What is the result after the execution of this code?

```
int k = 4, j = 10, *p = &k, *q = &j;  
    *p += *p + 55 + k - *q;  
printf("%d\n%d\n", k, *p + *q);
```

# Exercise

What is the result after the execution of this code?

```
i=1; j=2; k=3; k-=+i-j--;
```

```
i=1; j=2; k=3; k+=--i+--j;
```

```
i=1; j=2; k=3; k=-i++-j--;
```

```
i=1; j=2; k=3; k*=i+j--;
```

```
i=1; j=2; k=3; k%=--i+j++;
```

# Exercise

What is the result after the execution of this code? Explain your answer, please!

```
int k=1, j=0, i=1;  
while(i<=60)  
    j= (i++, k++);  
printf("j=%d\n",j);
```

# Exercise

What is the result after the execution of this code? Explain your answer, please!

```
int ax,b;  
ax=0x048;  
b=0x035;  
ax= ax | b;  
printf("b=%x, ax=%d\n",b, ax);
```

# Exercise

Find the mistakes!

```
#include <stdio>
int main
{
    int n;
    print(n=);
    scanf("%d",n);
    for(i=0;;i++);
        if (i=10) break;
        printf("%d ",i,i*i);
    return 0;
}
```

# Exercise

What is the result after the execution of this code?

```
int a, sum=0;
for (a=1;a<10;a++){
    if(!(a%2)){
        printf("%d ",a);
        sum=sum+a;
    }
}
printf("\n%d\n",sum);
```

# Exercise

What is the result after the execution of this code?

```
int a=10, b=20, *p=&a, *q=&b;  
*p +=(*q)++-15;  
printf("%d\t%d\n", a,*q);  
*q+=*p;  
printf("%d\t%d\n", a,*q);
```

# Exercise

What is the result after the execution of this code?

```
int a = 5, *p;
```

```
p=&a;
```

```
*p=a+*p-2;
```

```
printf("%d\n",a);
```



# Exercise

What is the result after the execution of this code?

```
int x = 7, *p, **q;  
p=&x;  
q=&p;  
x -= *p + **q;  
printf("%d\n",x);
```

# Exercise

What is the result after the execution of this code?

```
int i, j, k;  
i = -2; j = -3; k = 0;  
if(i == j || k)  
    k = -1 * (++i || j);  
else  
    k = -2 * (i && k);  
printf("%d\t%d\t%d\n", i, j, k);
```

# Exercise

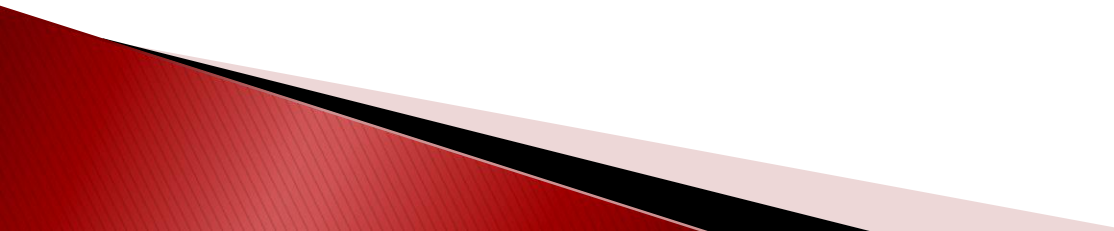
What is the result after the execution of this code?

```
int a = 0xEB, b = 0xAC, c = 0xEB;
```

```
c &= b << 3;
```

```
b ^= a >> 1;
```

```
printf("%X %X \n", c, b);
```



# Exercise

What is the result after the execution of this code?

```
int a=30, b=70, i, k, n=10;  
int t[10];  
    k=b-a+1;
```

```
    for (i=0;i<n;i++){  
        t[i]=rand()%k+a;  
        printf("%d\n",t[i]);
```

# Exercise

What is the result of this code?

```
double m [3][5]={5.5,7.9,5,7,34,10,11,-5,0,  
6.2,0,15,5,5,55.5,3};  
printf("%.3lf , %lf \n", m[2][1], m[1][3] );
```