

Aim of the course

Help the students to:

- understand basic concepts in Informatics,
- acquire basic programming skills.

<https://www.vecteezy.com>

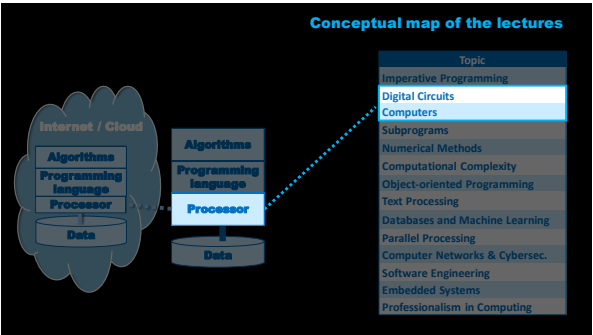
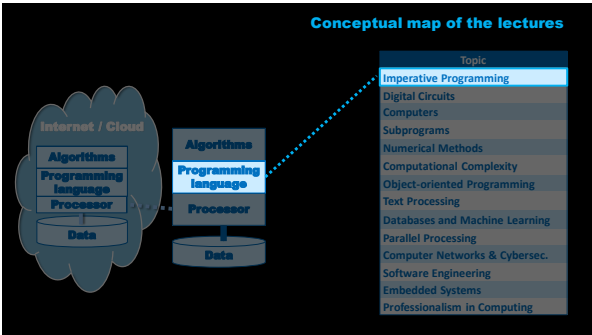
ACM Turing Award

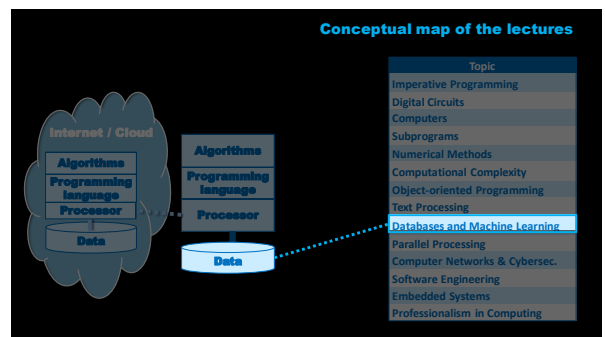
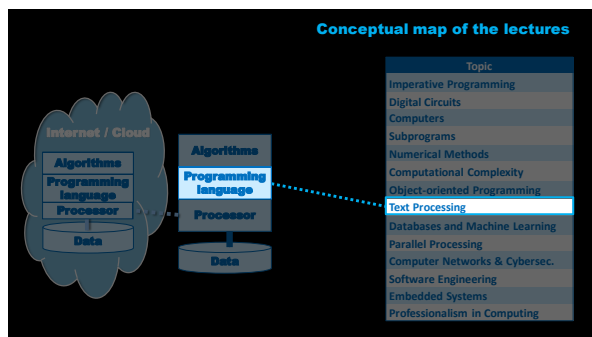
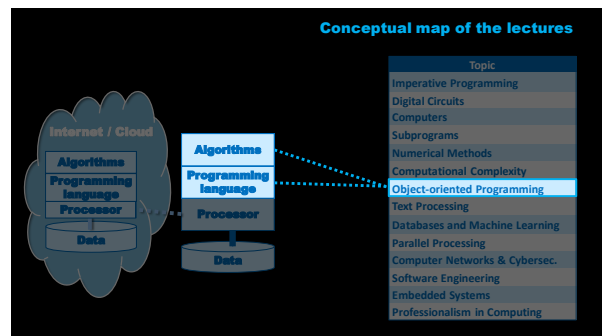
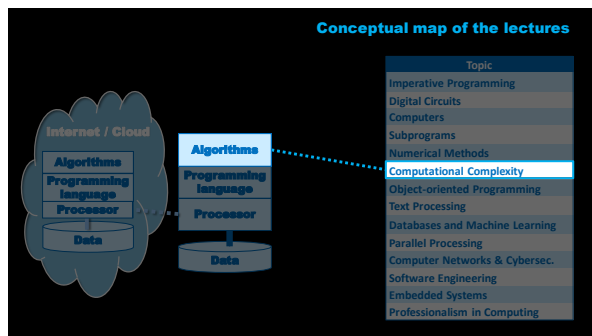
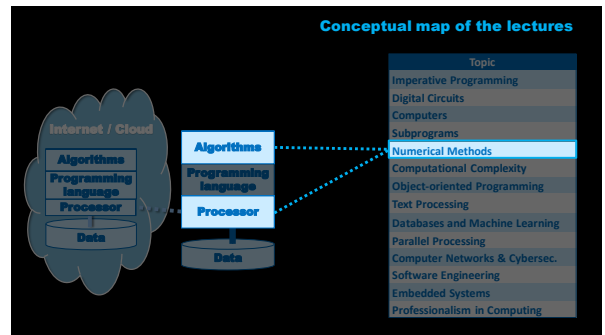
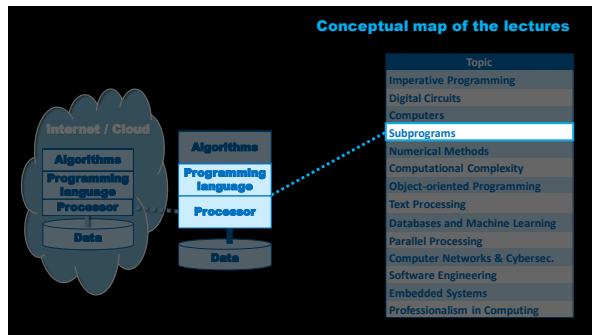
ACM = Association for Computing Machinery
Founded in 1947
About 110 000 members

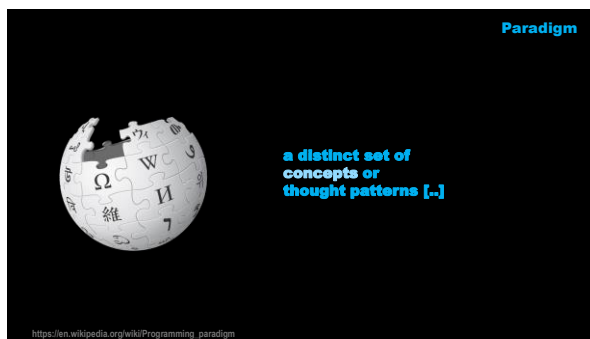
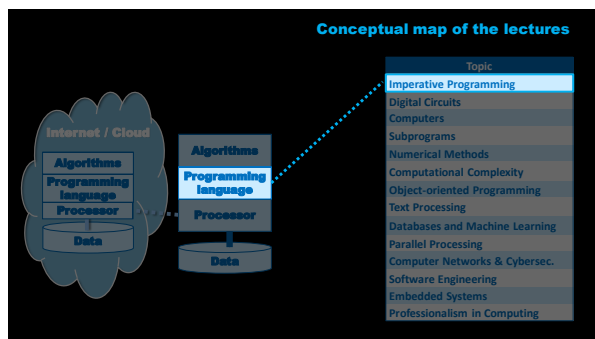
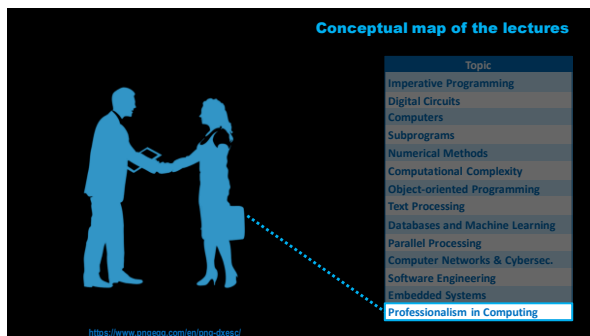
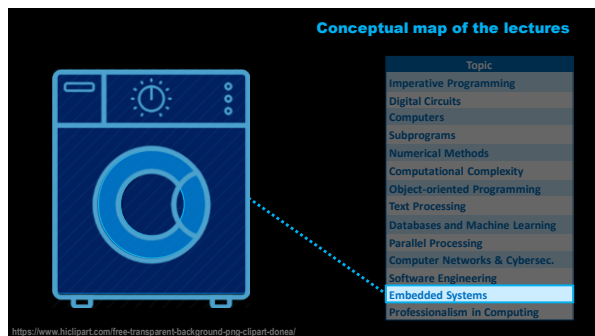
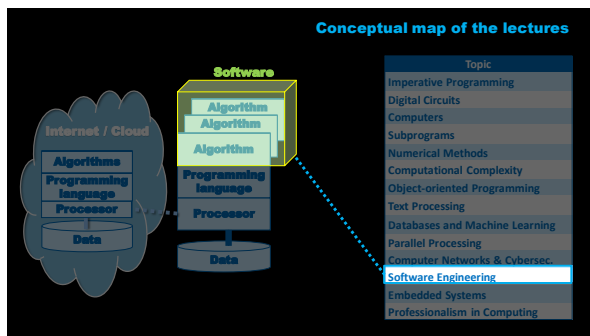
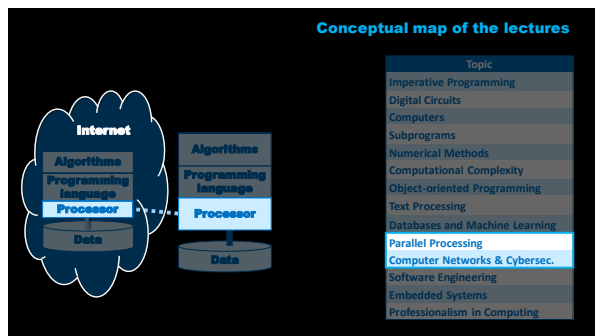
<https://www.acm.org/>

ACM Turing Award = „Nobel Prize of Computing“
Named after Alan Turing, a British mathematician
\$1 million (with financial support by Google)
Awarded since 1966


No.	Topic	Date	Timetable
1	Imperative Programming	2023-10-09	
2	Digital Circuits	2023-10-16	
3	Computers	2023-10-23	
4	Subprograms	2023-11-06	
5	Numerical Methods	2023-11-13	
6	Computational Complexity	2023-11-20	
7	Object-oriented Programming	2023-11-27	
8	Text Processing	2023-12-04	
9	Databases and Machine Learning	2023-12-11	
10	Parallel Processing	2023-12-18	
11	Computer Networks & Cybersecurity	2024-01-08	
12	Software Engineering	2024-01-15	
13	Embedded Systems	2024-01-22	
14	Professionalism in Computing	2024-01-29	







Programming paradigm



a distinct set of programming concepts or thought patterns [...]

https://en.wikipedia.org/wiki/Programming_paradigm

Introduction to Computing



Imperative programming

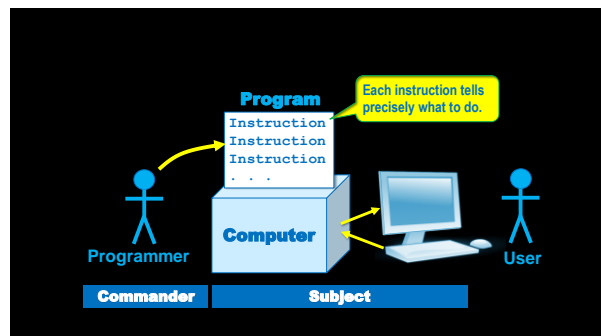
Jerzy Nawrocki
jerzy.nawrocki@put.poznan.pl
Faculty of Computing & Telecom.
Poznan University of Technology

Imperative programming




latin *imperare*
to command

Octavian Augustus
63 BC – 14 AD
The first emperor of the Roman Empire



Aim of this lecture



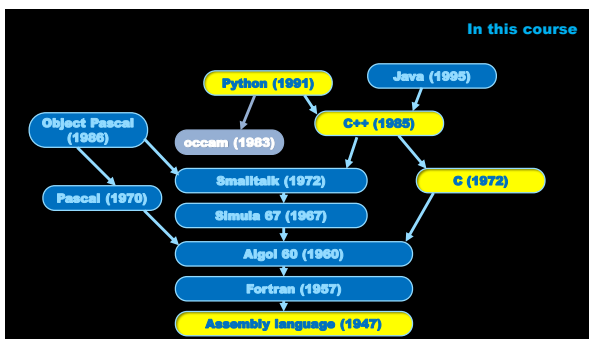
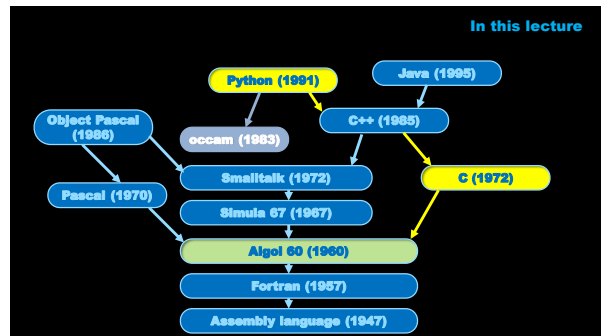
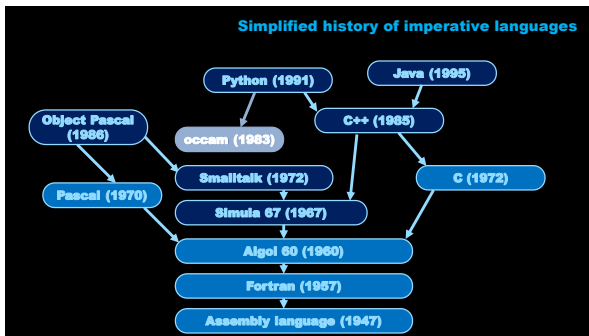
Help the students to understand the paradigm of imperative programming.

- Core of Python and C (only integers)
- Declarative programming – Prolog

<https://www.vecteezy.com>

Popularity of programming languages: TIOBE index

Mar 2022	Mar 2021	Change	Programming Language	Ratings
1	3	▲	Python	14.26%
2	1	▼	C	13.06%
3	2	▼	Java	11.19%
4	4		C++	8.66%
5	5		C#	5.92%
6	6		Visual Basic	5.77%
7	7		JavaScript	2.09%
8	8		PHP	1.92%



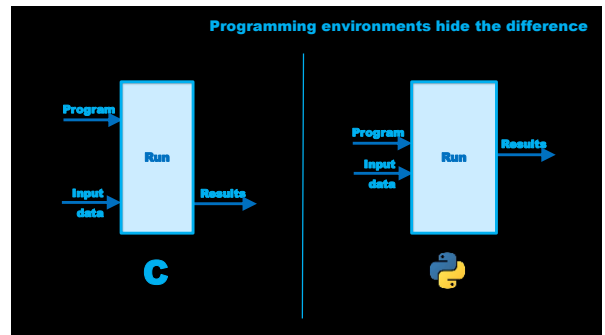
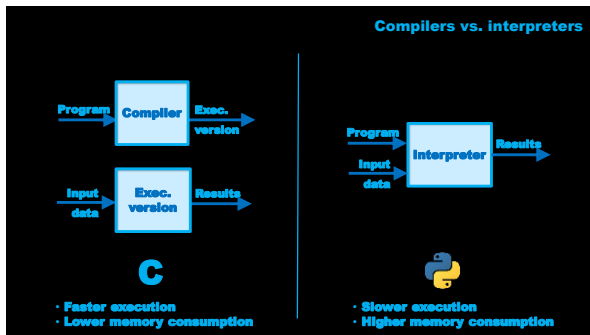
Two authors



Dennis M. Ritchie (1941 – 2011)
Bell Laboratories, USA
Main author of the C language
Turing Award 1983



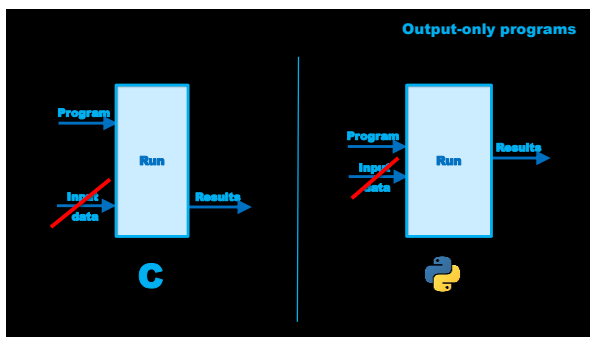
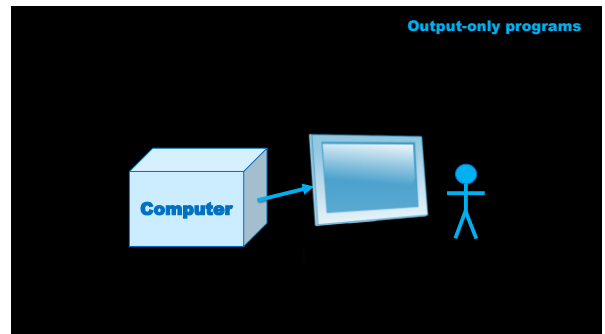
Guido van Rossum (born in 1956)
CWI, Amsterdam, the Netherlands
Author of the Python language
ACM Distinguished Engineer, 2006



Agenda

A photograph of a classical statue of Athena, standing and holding a spear and shield.

- **Very simple programs**
- Conditionals
- Iteration
- Simple functions
- Simple arrays
- Declarative programming



Extremely simple program

Two code snippets are shown. The first is Python code:

```
print("I don't know.")
```

 followed by the output `I don't know.`. The second is C code:

```
#include <stdio.h>
int main(void){
    printf("I don't know.\n");
}
```

Templates of simple programs



```
.....
```

C

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

Simple calculations



```
print(1+2*3)
7
```

C

```
#include <stdio.h>
int main(void){
    printf("%d\n", 1+2*3);
}
```

A more fancy version



```
print("1+2*3 =", 1+2*3)
1+2*3 = 7
```

C

```
#include <stdio.h>
int main(void){
    printf("1+2*3 = %d\n", 1+2*3);
}
```

Algebraic operators (integer numbers)

	C	Python
Addition	+	+
Subtraction	-	-
Multiplication	*	*
Division	/	//
Modulus	%	%
Power		**

Decimal digits 0 .. 9 and parenthesis (,)

Question



```
print("1+2*3 =", 1+2*3)
1+2*3 = 7
```

Why 7, not 9?


C

```
#include <stdio.h>
int main(void){
    printf("1+2*3 = %d\n", 1+2*3);
}
```


Priority of arithmetic operators

	C	Python	Priority
Addition	+	+	Low
Subtraction	-	-	Low
Multiplication	*	*	Medium
Division	/	//	Medium
Modulus	%	%	Medium
Power		**	High

Variables




```
a = 3
b = 4
c = a + b
print(c)
```




```
#include <stdio.h>
int main(void){
    int a, b, c;
    a = 3;
    b = 4;
    c = a + b;
    printf("%d\n", c);
}
```

7

Variable declaration




```
a = 3
b = 4
c = a + b
print(c)
```




```
#include <stdio.h>
int main(void){
    int a, b, c;
    a = 3;
    b = 4;
    c = a + b;
    printf("%d\n", c);
}
```

7

Quiz




```
a = 1
a = a + 4
print(a)
```



```
#include <stdio.h>
int main(void){
    int a;
    a = 1;
    a = a + 4;
    printf("%d\n", a);
}
```

5

How does assignment work?



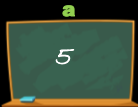
1

2

NameOfVariable = expression

```
a = 1
a = a + 4
```

How does assignment work?



1

2

NameOfVariable = expression

```
a = 1
a = a + 4
```

Useful shortcuts

```
a = a + 3
```

=

```
a += 3
```


```
var = var oper expr
```

=

```
var oper= expr
```


	C	Python
Addition	+	+
Subtraction	-	-
Multiplication	*	*
Division	/	//
Modulus	%	%
Power		**

Example

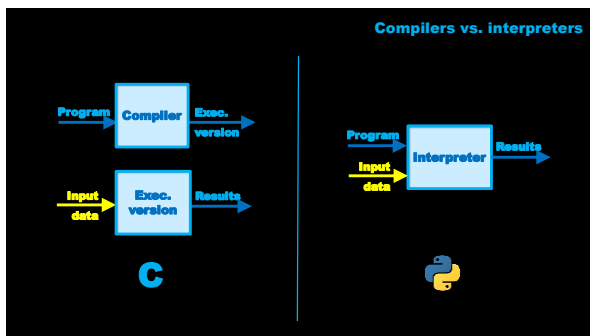
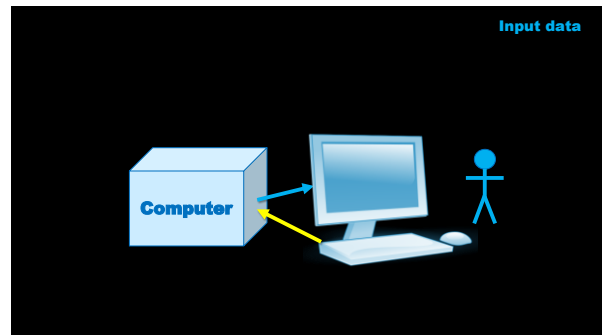


```
a = 1
a += 4
print(a)
```


5



```
#include <stdio.h>
int main(void){
    int a;
    a = 1;
    a += 4;
    printf("%d\n", a);
}
```




Quiz



```
a = int(input())
a = a + 4
print(a)
```

6



```
#include <stdio.h>
int main(void){
    int a;
    scanf("%d", &a);
    a = a + 4;
    printf("%d\n", a);
}
```

<https://www.tutorialspoint.com/codingground.htm>



Coding Platform For Your Website
Available for 75+ Programming Languages

Online Compilers and Interpreters

 Ada (Gnat)	 Algol 68	 Angular JS
 Assembly	 Awk	 Bash Shell
 Befunge	 Brain++	 C
 Chipmunk BASIC	 Clojure	 CSharp

<https://ideone.com/>

1) enter your source code or insert template or sample

2) Run your code here

justjoin.it

Najwięcej ofert pracy z widełkami

aplikuj teraz

Python 3

Run

<https://www.onlinegdb.com/>

```

1: //.....
2:
3: Welcome to GDB Online.
4: GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
5: C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML, CSS, JS
6: Code, Compile, Run and Debug online from anywhere in world.
7:
8: #include <stdio.h>
9:
10:
11: int main()
12: {
13:     printf("Hello World");
14:     return 0;
15: }
16:
17:

```

Agenda

- Very simple programs
- **Conditionals**
- Iteration
- Simple functions
- Simple arrays
- Declarative programming

Puzzle 5

x

3

```

-3
x = int(input())
if x < 0:
    x = - x
print(x)
3

```

```

#include <stdio.h>
int main(void){
    int x;
    scanf("%d", &x);
    if ( x < 0 ) {
        x = - x ;
    }
    printf("%d\n", x);
}

```

Conditional statement

< > <= >= == !=

```

if condition :
    instruction

```

Indentation!

```

if ( condition ) {
    instruction
}

```

Conditional statement

< > <= >= == !=

```

if condition :
    instruction_1
else:
    instruction_2

```

Indentation!

```

if ( condition ) {
    instruction_1
}
else {
    instruction_2
}

```

Compound conditions

```

c1 && c2
c1 || c2
! c2

```

```

c1 and c2
c1 or c2
not c2

```

c1	c2	c1 && c2
false	false	false
false	true	false
true	false	false
true	true	true

Compound conditions

C

```
c1 && c2
c1 || c2
! c2
```

Python

```
c1 and c2
c1 or c2
not c2
```

c1	c2	c1 c2
false	false	false
false	true	true
true	false	true
true	true	true

Compound conditions

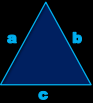
C

```
c1 && c2
c1 || c2
! c2
```

Python

```
c1 and c2
c1 or c2
not c2
```


c2	! c2
false	true
true	false
false	true
true	false



```
a = int(input())
b = int(input())
c = int(input())
if a==b and b==c :
    print("Equilateral")
else:
    print("Not equilateral")
```

C

```
#include <stdio.h>
int main(void){
    int a, b, c;
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &c);
    if (a==b && b==c){
        printf("Equilateral\n");
    }
    else {
        printf("Not equilateral\n");
    }
}
```



Agenda

- Very simple programs
- Conditionals
- Iteration
- Simple functions
- Simple arrays
- Declarative programming

Iteration

Python

```
while condition :
    instruction
```

C

```
while ( condition ) {
    instruction
}
```

Indentation!

Greatest Common Divisor


a	b
15	10
5	5

Python

```
a = int(input())
b = int(input())
while a != b:
    if a > b:
        a = b
    else:
        b = a
print(a)
```

C

```
int main(void) {
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    while ( a != b ){
        if(a > b){
            a = b;
        }
        else{
            b = a;
        }
    }
    printf("%d\n", a);
}
```



Agenda

- Very simple programs
- Conditionals
- Iteration
- **Simple functions**
- Simple arrays
- Declarative programming

Functions in mathematics

Function name: $\sin(x)$
Parameter (argument): $\cos(x)$
...
 $\text{power}(b, e) = b^e$

Greatest Common Divisor

```
Python
a = int(input())
b = int(input())
while a != b:
    if a > b:
        a = b
    else:
        b = a
print(a)
```

```
C
int main(void) {
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    while (a != b) {
        if (a > b) {
            a = b;
        } else {
            b = a;
        }
    }
    printf("%d\n", a);
}
```

If GCD(a,b) was already available ...

```
Python
x = int(input())
y = int(input())
print(GCD(x, y))
```

```
C
#include <stdio.h>
int main(void) {
    int x, y;
    scanf("%d", &x);
    scanf("%d", &y);
    printf("%d", GCD(x,y));
}
```

Wrapping up instructions into a function

```
Python
a = int(input())
b = int(input())
while a != b:
    if a > b:
        a = b
    else:
        b = a
print(a)
```

Function name: def GCD(a, b):

```
Python
def GCD(a, b):
    while a != b:
        if a > b:
            a = b
        else:
            b = a
    return a
```

Wrapping up instructions into a function

```
C
int main(void) {
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    while (a != b) {
        if (a > b) {
            a = b;
        } else {
            b = a;
        }
    }
    printf("%d\n", a);
}
```

Function name: $\text{int GCD(int a, int b) \{}$

```
C
int GCD(int a, int b) {
    while (a != b) {
        if (a > b) {
            a = b;
        } else {
            b = a;
        }
    }
    return a;
}
```

a	b

x	y

```

6
9

def GCD(a, b):
    while a != b:
        if a > b:
            a -= b
        else:
            b -= a
    return a

x = int(input())
y = int(input())
print(GCD(x, y))

5
    
```

```

#include <stdio.h>

int GCD(int a, int b) {
    while (a != b) {
        if (a > b) {
            a = b;
        } else {
            b = a;
        }
    }
    return a;
}

int main(void) {
    int x, y;
    scanf("%d", &x);
    scanf("%d", &y);
    printf("%d", GCD(x, y));
}
    
```

Structure of a function definition

```

def funName ( params ) :
    localDeclarations
    instruction
    ...
    instruction
    return expression
    
```

```

resultType funName ( params ) {
    localDeclarations
    instruction
    ...
    instruction
    return expression;
}
    
```

Agenda

- Very simple programs
- Conditionals
- Iteration
- Simple functions
- Simple arrays

- Declarative programming

Array

```

write(5, 9)    Num[5] = 9;
read(5)        . = Num[5];
    
```

Index	Name
0	Num
1	
2	
3	
4	
5	9
6	
7	
8	
9	

Variable declaration in C

x1

x2

Simple variable:

- Name
- Type

```

int x1;
int x2;

int x1, x2;
    
```

Array declaration in Pascal

x1

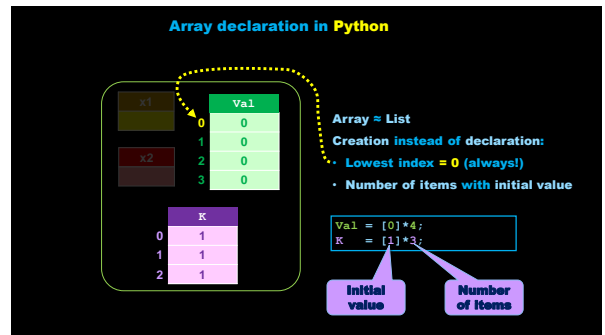
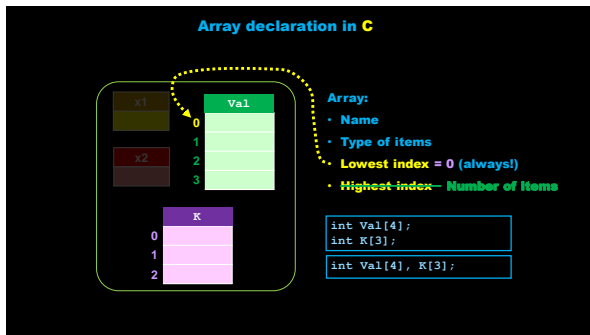
x2

Array:

- Name
- Type of items
- Lowest index
- Highest index

```

var
  Val: array[0..3] of integer;
    
```



Write a C program that reads n ($n \leq 10$) then n integers x_1, \dots, x_n and displays the sequence x_i in the reverse order.

n	i	x
3	0	
10	1	
20	2	
30	3	
	4	
	5	
	6	
	7	
	8	
	9	

```
#include <stdio.h>
int main(void){
    int n, x[10], i;
    scanf("%d", &n);
    i=0;
    while (i < n){
        scanf("%d", &x[i]);
        i+=1;
    }
    i-=1;
    while (i >= 0){
        printf("%d ", x[i]);
        i-=1;
    }
}
```

Write a C program that reads n ($n \leq 10$) then n integers x_1, \dots, x_n and displays the sequence x_i in the reverse order.

n	i	x
3	0	
10	1	
20	2	
30	3	
	4	
	5	
	6	
	7	
	8	
	9	

```
#include <stdio.h>
int main(void){
    int n, x[10], i;
    scanf("%d", &n);
    i=0;
    while (i < n){
        x[i]= int(input());
        i+=1;
    }
    i-=1;
    while (i >= 0){
        printf("%d ", x[i]);
        i-=1;
    }
}
```

Write a C program that reads n ($n \leq 10$) then n integers x_1, \dots, x_n and displays the sequence x_i in the reverse order.


n	i	x
3	0	10
10	1	20
20	2	30
30	3	
	4	
	5	
	6	
	7	
	8	
	9	

```
#include <stdio.h>
int main(void){
    int n, x[10], i;
    scanf("%d", &n);
    i=0;
    while (i < n){
        scanf("%d", &x[i]);
        i+=1;
    }
    i-=1;
    while (i >= 0){
        printf("%d ", x[i]);
        i-=1;
    }
}
```

Write a C program that reads n ($n \leq 10$) then n integers x_1, \dots, x_n and displays the sequence x_i in the reverse order.

n	i	x
3	0	10
10	1	20
20	2	30
30	3	
	4	
	5	
	6	
	7	
	8	
	9	

```
#include <stdio.h>
int main(void){
    int n, x[10], i;
    scanf("%d", &n);
    i=0;
    while (i < n){
        scanf("%d", &x[i]);
        i+=1;
    }
    i-=1;
    while (i >= 0){
        printf("%d ", x[i]);
        i-=1;
    }
}
```



Agenda

- Very simple programs
- Conditionals
- Iteration
- Simple functions
- Simple arrays
- Declarative programming

```
:- initialization(main).
father(elizabeth2, george6).
father(george6, george5).
father(george5, edward7).
father(edward7, albertCoburg).
father(aliceHesse, albertCoburg).
mother(elizabeth2, elizabethBowesLyon).
mother(george6, maryTeck).
mother(george5, alexandraDenmark).
mother(edward7, queenVictoria).
mother(aliceHesse, queenVictoria).
parent(X,Y):- father(X,Y); mother(X,Y).
grandma(X,Y):- parent(X,Z), mother(Z,Y).
grandpa(X,Y):- parent(X,Z), father(Z,Y).
greatGrandma(X,Y):- parent(X,Z), grandma(Z,Y).
greatGrandpa(X,Y):- parent(X,Z), grandpa(Z,Y).
great2Grandma(X,Y):- parent(X,Z), greatGrandma(Z,Y).
great2Grandpa(X,Y):- parent(X,Z), greatGrandpa(Z,Y).
main:- great2Grandma(elizabeth2,Y),
       write(Y), nl.
```

Prolog

Facts

```
:- initialization(main).
father(elizabeth2, george6).
father(george6, george5).
father(george5, edward7).
father(edward7, albertCoburg).
father(aliceHesse, albertCoburg).
mother(elizabeth2, elizabethBowesLyon).
mother(george6, maryTeck).
mother(george5, alexandraDenmark).
mother(edward7, queenVictoria).
mother(aliceHesse, queenVictoria).
parent(X,Y):- father(X,Y); mother(X,Y).
grandma(X,Y):- parent(X,Z), mother(Z,Y).
grandpa(X,Y):- parent(X,Z), father(Z,Y).
greatGrandma(X,Y):- parent(X,Z), grandma(Z,Y).
greatGrandpa(X,Y):- parent(X,Z), grandpa(Z,Y).
great2Grandma(X,Y):- parent(X,Z), greatGrandma(Z,Y).
great2Grandpa(X,Y):- parent(X,Z), greatGrandpa(Z,Y).
main:- great2Grandma(elizabeth2,Y),
       write(Y), nl.
```


Prolog

Inference rules

```
:- initialization(main).
father(elizabeth2, george6).
father(george6, george5).
father(george5, edward7).
father(edward7, albertCoburg).
father(aliceHesse, albertCoburg).
mother(elizabeth2, elizabethBowesLyon).
mother(george6, maryTeck).
mother(george5, alexandraDenmark).
mother(edward7, queenVictoria).
mother(aliceHesse, queenVictoria).
parent(X,Y):- father(X,Y); mother(X,Y).
grandma(X,Y):- parent(X,Z), mother(Z,Y).
grandpa(X,Y):- parent(X,Z), father(Z,Y).
greatGrandma(X,Y):- parent(X,Z), grandma(Z,Y).
greatGrandpa(X,Y):- parent(X,Z), grandpa(Z,Y).
great2Grandma(X,Y):- parent(X,Z), greatGrandma(Z,Y).
great2Grandpa(X,Y):- parent(X,Z), greatGrandpa(Z,Y).
main:- great2Grandma(elizabeth2,Y),
       write(Y), nl.
```


Prolog

Goal / question




Summary

Core of imperative programming



```
variable = int(input())
print(expression)
variable = expression
if condition:
    instruction
while condition:
    instruction
def name(params):
    instructions
    return expression
ArrayName[expression]
```

Watch Indentation!



```
scanf("%d", &variable);
printf("%d\n", expression);
variable = expression;
if ( condition ) {
    instruction }
while ( condition ) {
    instruction }
int name(params){
    instructions
    return expression; }
ArrayName[expression]
```


© Jerzy Nawrocki, Introduction to Computing

Recommended readings



B. Kernighan, D. Ritchie,
The C Programming Language.



<https://www.w3schools.com/c/>
<https://www.w3schools.com/python/>

<https://en.vetores.org/>

Grading rules

- Individual test (optional)
- Team contest (optional)
- Exam (February – compulsory)

Important:

There is one grade for the whole course.

Prize of the team contest: Guaranteed grade



Guaranteed grade:

- you can accept the grade or
- try to improve it by taking the exam in February

Important:

To get the guaranteed grade you have to pass the **Individual test**.

<https://www.clipartmax.com/>

Team contest



Each team consists of up to **4 people**.

<https://www.hiclipart.com/free-transparent-background-png-clipart-mqjw/download>

No.	Topic	Date
1	Imperative Programming	2023-10-09
2	Digital Circuits	2023-10-16
3	Computers	2023-10-23
4	Subprograms	2023-11-06
5	Numerical Methods	2023-11-13
6	Computational Complexity	2023-11-20
7	Object-oriented Programming	2023-11-27
8	Text Processing	2023-12-04
9	Databases and Machine Learning	2023-12-11
10	Parallel Processing	2023-12-18
11	Computer Networks & Cybersecurity	2024-01-08
12	Software Engineering	2024-01-15
13	Embedded Systems	2024-01-22
14	Professionalism in Computing	2024-01-29

Timetable
Tentative
Individual Test (topics 1-8) 2023-12-21
Team Contest (topics 1-11) 2024-01-11

