Introduction to Programming

Labor 02

Basic Information

- **Type of course:** Labor
- **Subject code:** INBPA0104-17
- **Credit:** 3
- https://elearning.unideb.hu/course/view.php?id=9468
- **Password/Enrollment key:** IntroProg2022
- **Lecturers:**
  - Piroska Biró, PhD
  - Anikó Apró

Contact & Office Hours

- **Piroska Biró, PhD**
  - Office Hours: IK-227 or Online -> MS Teams
    - Tuesday 13:00-14:00
    - Wednesday 14:00-15:00
  - E-mail:
    - biro.piroska@inf.unideb.hu
- **Anikó Apró**
  - Office Hours: IK-229 or Online -> MS Teams
    - Monday 11:00-12:00
    - Tuesday 11:30-12:30
  - E-mail:
    - aniko.apro@inf.unideb.hu

Requirements

- **Attendance and Participation:**
  - In every labor there will be an attendance sheet.
  - **Maximum three absences** are allowed in labor.
  - **Maximum 15 minutes late arrival** is accepted in labor.

Assessment and grading:
  - Midterm max. 100 points must be achieved **min. 50 points**
  - Endterm max. 100 points must be achieved **min. 50 points**
  - Midterm + Endterm max. 200 points must be achieved **min. 100 points**
- **Assessment: Practical mark**
  - To calculate the Final Grade the following formula and table should be used.
  - Final Point= (Midterm + Endterm)/2

Grade	Final Point
5	90 – 100
4	80 – 89
3	65 – 79
2	50 – 64
1	0 – 49

Revision – Compiling C Program

- Regular source code files.
  - hello.c
- Header files.
  - stdio.h or math.h
- Object files.
  - hello.o
- Binary executables.
  - a.out or hello

Revision

- **mkdir** lab02
- **cd** lab02

- Write a C program that should write a single line to the standard output containing the string „Good morning!/Good afternoon!".

prog1.c

```
$gcc prog1.c -o prog1
$./prog1
```

```
#include <stdio.h>
int main() {
	printf("Good morning!\n");
	//printf("Good afternoon!\n");
	return 0;
}
```

Syntax & Semantics

- **Syntax** – the grammatical rules and structural patterns governing the ordered use of appropriate words and symbols for issuing commands, writing code, etc., in a particular software application or programming language.
- **Semantics** – the meaning, or an interpretation of the meaning, of a word, sign, sentence, etc.

Syntax – Examples

- **Syntax** is about the structure or the grammar of the language.
- It answers the question: how do I construct a valid sentence?
- All languages, even English and other human (aka "natural") languages have grammars, that is, rules that define whether or not the sentence is properly constructed.

- Here are some C language syntax rules:
  - separate statements with a semicolon
  - enclose the conditional expression of an **if** statement inside parentheses
  - group multiple statements into a single statement by enclosing in curly brackets
  - data types and variables must be declared before the first executable statement (this feature has been dropped in C99. C99 and latter allow mixed type declarations.)

Semantics – Examples

- **Semantics** is about the meaning of the sentence.
- It answers the questions: is this sentence/program valid?
- If so, what does the sentence/program mean?

For example:

- x++;   // increment

Variables

- differences between small and capital letter
- using **&** operator we can refer to the address of (the variable) **a**, **&a**
- **char, int**: store the integer numbers
- **char**: store characters
- **float, double**: store the real numbers

Definition of the variables

- **int** a, b=3;
- **float** b1, b2=2.5;
- **double** x, y=3.14;

Types of data

`char`	8	$-128 .. 127$
`unsigned char`	8	$0 .. 255$
`short`	16	$-32768 .. 32767$
`unsigned short`	16	$0 .. 65535$
`int`	16	$-32768 .. 32767$
`int`	32	$-2147483648 .. 2147483647$
`unsigned int`	16	$0 .. 65535$
`unsigned int`	32	$0 .. 4294967295$
`long`	32	$-2147483648 .. 2147483647$
`unsigned long`	32	$0 .. 4294967295$
`float`	32	$3.4 \cdot 10^{-38} .. 3.4 \cdot 10^{38}$
`double`	64	$1.7 \cdot 10^{-308} .. 1.7 \cdot 10^{308}$
`long double`	80	$3.4 \cdot 10^{-4932} .. 1.1 \cdot 10^{4932}$

Format Specifiers

Type	Format specifiers
char	%c
int	%d or %i (base 10), %o (base 8), %x, %X (base 16)
unsigned int	%u
short int	%hd or %hi
unsigned short int	%hu
long int	%ld or %li
unsigned long int	%lu
float	%f
double	%lf
long double	%Lf
string	%s

Keywords/Reserved words in C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Arithmetic operators

Basic assignment		a=b
Addition		a+b
Subtraction		a−b
Unary plus		+a
Unary minus		−a
Multiplication		a*b
Division		a/b
Modulo (integer remainder)		a%b
Increment	prefix	++a
	suffix	a++
Decrement	prefix	−−a
	suffix	a−−

Comparison operators (relational operators)

Equal to	a==b
Not equal to	a!=b
Greater than	a>b
Less than	a<b
Greater than or equal to	a>=b
Less than or equal to	a<=b

```
a == 5
    /* Does NOT assign five to a. Rather, it checks to see if a equals 5.*/
```

Logical operators

Logical negation (NOT)	!a
Logical AND	a&&b
Logical OR	a||b

Example:

!5, !!5, 5&&6, 0&&13, 0||12;
0 and 1 logical value!!!

Bitwise operators

Bitwise NOT	~a
Bitwise AND	a&b
Bitwise OR	a|b
Bitwise XOR	a^b
Bitwise left shift	a<<b
Bitwise right shift	a>>b

Compound assignment operators

Addition assignment	a += b	a = a + b
Subtraction assignment	a −= b	a = a − b
Multiplication assignment	a *= b	a = a * b
Division assignment	a /= b	a = a / b
Modulo assignment	a %= b	a = a % b
Bitwise AND assignment	a &= b	a = a & b
Bitwise OR assignment	a |= b	a = a | b
Bitwise XOR assignment	a ^= b	a = a ^ b
Bitwise left shift assignment	a <<= b	a = a << b
Bitwise right shift assignment	a >>= b	a = a >> b

Number systems

- Decimal number system
  - the concept of place value
- Generally the numbers can be described in the following form:

$$a_n a_{n-1} \ldots a_1 a_0 a_{-1} a_{-2} \ldots a_{-m}$$

$\sum_{i=-m}^{n} a_i \cdot 10^i$, when $0 \leq a_i < 10$.

- Optional in the p base (p>1) number system the used digits 0, 1, …, p−1, the place values are the powers of number p:

$\sum_{i=-m}^{n} a_i \cdot p^i$, when $0 \leq a_i < p$.

Example: 123,45

Number systems

- Binary (base two) number system
  - digits: 0, 1
- Ternary (base three) number system
  - digits: 0, 1, 2

…
- Octonary (base eight) number system
  - digits: 0, 1, 2,…, 7

…
- Decimal (base ten) number system/ten base
  - digits: 0, 1, 2,…, 9

…
- Hexadecimal (base sixteen) number system
  - digits: 0, 1, 2, …, 9, A, B, C, D, E, F

Binary p = 2	Ternary p = 3	Quinary p = 5	Octonary p = 8	Decimal p = 10	Duodecimal p = 12	Hexadecimal p = 16
0	0	0	0	0	0	0
1	1	1	1	1	1	1
10	2	2	2	2	2	2
11	10	3	3	3	3	3
100	11	4	4	4	4	4
101	12	10	5	5	5	5
110	20	11	6	6	6	6
111	21	12	7	7	7	7
1000	22	13	10	8	8	8
1001	100	14	11	9	9	9
1010	101	20	12	10	a	A
1011	102	21	13	11	b	B
1100	110	22	14	12	10	C
1101	111	23	15	13	11	D
1110	112	24	16	14	12	E
1111	120	25	17	15	13	F
10000	121	26	20	16	14	10

Number systems base names

- 2 binary
- 3 ternary
- 4 quaternary
- 5 quinary
- 6 senary
- 7 septenary
- 8 octonary
- 9 nonary
- 10 decimal
- 11 undenary
- 12 duodecimal
- 13 tridecimal
- 14 quattuordecimal
- 15 quindecimal
- 16 sexadecimal

Convert to base 10 (p->10)

Generally:

$$a_n a_{n-1} \ldots a_0 . a_{-1} a_{-2} \ldots a_{-m\ p} =$$
$$a_n \cdot\ p^n + a_{n-1} \cdot\ p^{n-1} + \cdots + a_0 \cdot\ p^0 + a_{-1} \cdot\ p^{-1} +$$
$$+ a_{-2} \cdot\ p^{-2} + \cdots a_{-m} \cdot\ p^m$$
$$0 \leq a_i < p$$

For example:

$$263.15_7 = 2 \cdot 7^2 + 6 \cdot 7^1 + 3 \cdot 7^0 + 1 \cdot 7^{-1} + 5 \cdot 7^{-2} =$$
$$= 98 + 49 + 3 + \frac{1}{7} + \frac{5}{7^2} = 143 \frac{12}{49}_{10}$$

Exercise

- Convert each of the following different base representation to its equivalent base ten form:

1. $1011100.101_2 =$
2. $1221.12_3 =$
3. $152.43_6 =$
4. $173.104_8 =$
5. $841.47_9 =$
6. $13A.2F_{16} =$

Convert from base 10 to different base number representations (10->p)

$113.45_{(10}$

113	2
56	1
28	0
14	0
7	0
3	1
1	1
0	1

0	.45	2
0	.90	
1	.8	
1	.6	
1	.2	
0	.4	
0	.8	
1	.6	
1	.2	
0	.4	
0	.8	

$1100001.0111001100_{(2}$

Exercise

- Convert each of the following base ten representation to its equivalent **binary** form:
  1. $962_{10} =$
  2. $3241_{10} =$
  3. $871.64_{10} =$
  4. $1322.181_{10} =$
- Convert each of the following ten base representation to its equivalent **octonary** form:
  1. $51_{10}=$
  2. $718_{10}=$
  3. $417.18_{10}=$
  4. $791.27_{10}=$
- Convert each of the following ten base representation to its equivalent **hexadecimal** form:
  1. $334_{10}=$
  2. $8191_{10}=$
  3. $218.2_{10}=$
  4. $245.17_{10}=$

Exercise – Homework!

- Convert each of the following ten base representation to its equivalent **binary** (**base 2**) form:
  1. $1862_{10} =$
  2. $93281_{10} =$
  3. $39871.64_{10} =$
  4. $49322.1813_{10} =$
- Convert each of the following ten base representation to its equivalent **base seven** (septenary) form:
  1. $1951_{10}=$
  2. $82718_{10}=$
  3. $417.18_{10}=$
  4. $13791.27_{10}=$
- Convert each of the following ten base representation to its equivalent **base nine** (nonary) form:
  1. $2334_{10}=$
  2. $83191_{10}=$
  3. $218.92_{10}=$
  4. $5245.67_{10}=$

Connection between the octal and the binary system

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Connection between the hexadecimal and the binary system

Binary	Hexadecimal	Binary	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Exercise – Homework!

- Convert each of the following binary representation to its equivalent octal and hexadecimal form, and each of the following hexadecimal representation to its equivalent binary and octal form:

1. $1110\ 1001\ 1100\ 0011_2 =$
2. $1011\ 0111\ 0101\ 0100_2 =$
3. $1000\ 1101\ 1111\ 0011\ 1101_2 =$
4. $1010\ 1011\ 0011\ 1110\ 0001\ 0101_2 =$
5. $3BCF_{16} =$
6. $BF29_{16} =$
7. $48C5_{16} =$
8. $63AE_{16} =$

Webpages – Practicing – Converters

- http://planetcalc.com/862/
- http://planetcalc.com/2095/