

CPE101 Programming Languages I

Week 8

Variable Types in C Language and Basic Input/Output Operations

Assoc. Prof. Dr. Caner ÖZCAN

Basic Data Types in C Language

- ▶ Variables and data types to be used in C language must be declared in the program beforehand.
- ▶ Digital Data Types
 - a) Integer Data Types
 - b) Fractional Data Types
 - `int` – integers
 - `float` – float numbers
 - `double` – longer and very sensitive float numbers
 - `char` - characters

Integers

- ▶ Represent integers
 - Both negative and positive integers
- ▶ Expression of integer type in C :

int

- ▶ Example:

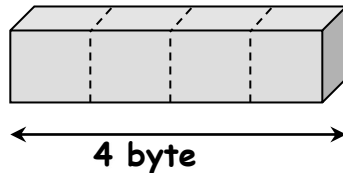
```
int toplam;    /* signed integer */  
  
toplam = 100;  /* can be positive */  
toplam = -20; /* can be negative */
```

```
int toplam = 32000; /* initialization can    */  
                  /* be made when definition */
```

Integers

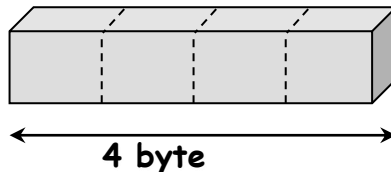
- ▶ Integer qualifiers: **long**, **short**, or **unsigned**
- ▶ Integer sizes vary according to the qualifiers.
- ▶ The default integer size depends on the machine operating system.

int



from -2.147.483.648 to
2.147.483.647 (total number
4.294.967.296)

**unsigned
int**



from 0 to 4,294,967,295
(total number 4,294,967,296)

Fractional Numbers- float

- ▶ It refers to the actual number (with comma section)
 - Can be negative and positive
- ▶ Expression of float type in C :
float
- ▶ Example:

```
float f;  
  
f = 0.12;    /* can be positive */  
f = -245.56; /* can be negative */
```

```
float f = 4.567; /* initialization can */  
                /* be made when definition */
```

Longer Fractional Numbers - double

- ▶ Standard "double precision floating point" (real) numbers.
 - such as float, but is much larger and precision.
- ▶ Expression of double type in C:
double
- ▶ Example:

```
double d;
```

```
d = 3.12E+5;    /* 312000.0 */
```

```
d = -45.678;    /* negative */
```

```
double d = 4.567; /* initialization */
```

Character - char

- ▶ It refers to a single character
 - Characters
 - Uppercase and lowercase letters of the alphabet
 - 10 numbers from 0 to 9
 - Special symbols such as +#@½%&\$.*?!£'=-:/*^(){}[]~;;,<>
- ▶ Characters used between quotation marks
 - for example 'A'
- ▶ Expression of char type in C :

char

```
char c;
```

```
c = 'A';    /* Letter A */
```

```
c = '9';    /* Number 9 */
```

```
char c = 'c'; /* initialization */
```

char

- ▶ Actually, the characters represent a natural number with 1 byte
 - char variable takes place 1 byte in memory
- ▶ Characters (char variables) values in ASCII table...
 - 'A' ASCII value is 65
 - 'B' ASCII value is 66
 - '0' ASCII value is 48
 - '1' ASCII value is 49
 - <http://www.asciitable.com/>

ASCII Tablosu

Decimal Hex Char			Decimal Hex Char			Decimal Hex Char			Decimal Hex Char		
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Special Characters

- ▶ Characters are shown in single quotes
 - How we will show quotation marks?

```
char c;  
c = '';
```



```
char c;  
c = '\\';
```



- ▶ If backslash(\) is used before a character, this case is called as **Escape Sequence**.
 - It destroys the meaning of the character after coming from it.

Escape Sequence

- ▶ It consists of backslash (\) and one character. The compiler gives the sign to be perceived as normal the next character.
- ▶ Favorites
 - \n go to the next line
 - \t move to the next tab
 - \r takes per line
 - \\ backslash character
 - \' single quotes
 - \" double quotes

Summary

- ▶ Integers (signed and unsigned)
 - `char` – 1 byte
 - Also used to store ASCII characters..
 - `short` – 2 byte
 - `int` – 4 byte
 - `long` – 4 veya 8 byte
- ▶ Real numbers (just signed)
 - `float` – 4 byte
 - `double` – 8 byte

Data Types and Features

Data Type	Size	Range
char	1 byte	-128 : 127
unsigned char	1 byte	0 : 255
short	2 byte	-32768 : 32767
unsigned short	2 byte	0 : 65535
int	4 byte	-2147483648 : 2147483647
unsigned int	4 byte	0 : 4294967295
float (7 precision)	4 byte	1.175494e-38 : 3.402823e+38
double (16 precision)	8 byte	2.225074e-308 : 1.797693e+308

Basic Writing Characteristics of C Language

- ▶ Program writing is in the form of certain patterns and blocks.
- ▶ The blocks are created by brackets {}.
- ▶ Commands can be written to the same or lower bottom line. maximum of 1023 characters can be written on one line.
- ▶ All commands ends with semicolon (;).
- ▶ Semicolons is not used after the phrase started block.
- ▶ All variables used in the program and data types are defined.
- ▶ The libraries containing the commands to be used in the program must be activated / called.

Structure of C Language

- ▶ **Program title:** The section contains the description about the program.

/* description or program title */

- ▶ **Definition and Declaration Part:** This section includes preprocessor commands, variables and structure identification, notification, such as a fixed value assignment.

a) include: used to call the library.

#include < *library name* >

stdio.h: standart input/output

conio.h: dos supported input/output

math.h: mathematical functions

stdlib.h: transform, sort, search, and so on.

Structure of C Language

► Definition and Declaration Part:

- b) define:** Command that allows the transfer some expressions or constants to the symbolic name.

#define symbolic_name equivalent_expression

- c) Variable definition:** All variables in C are reported as the name and data type..

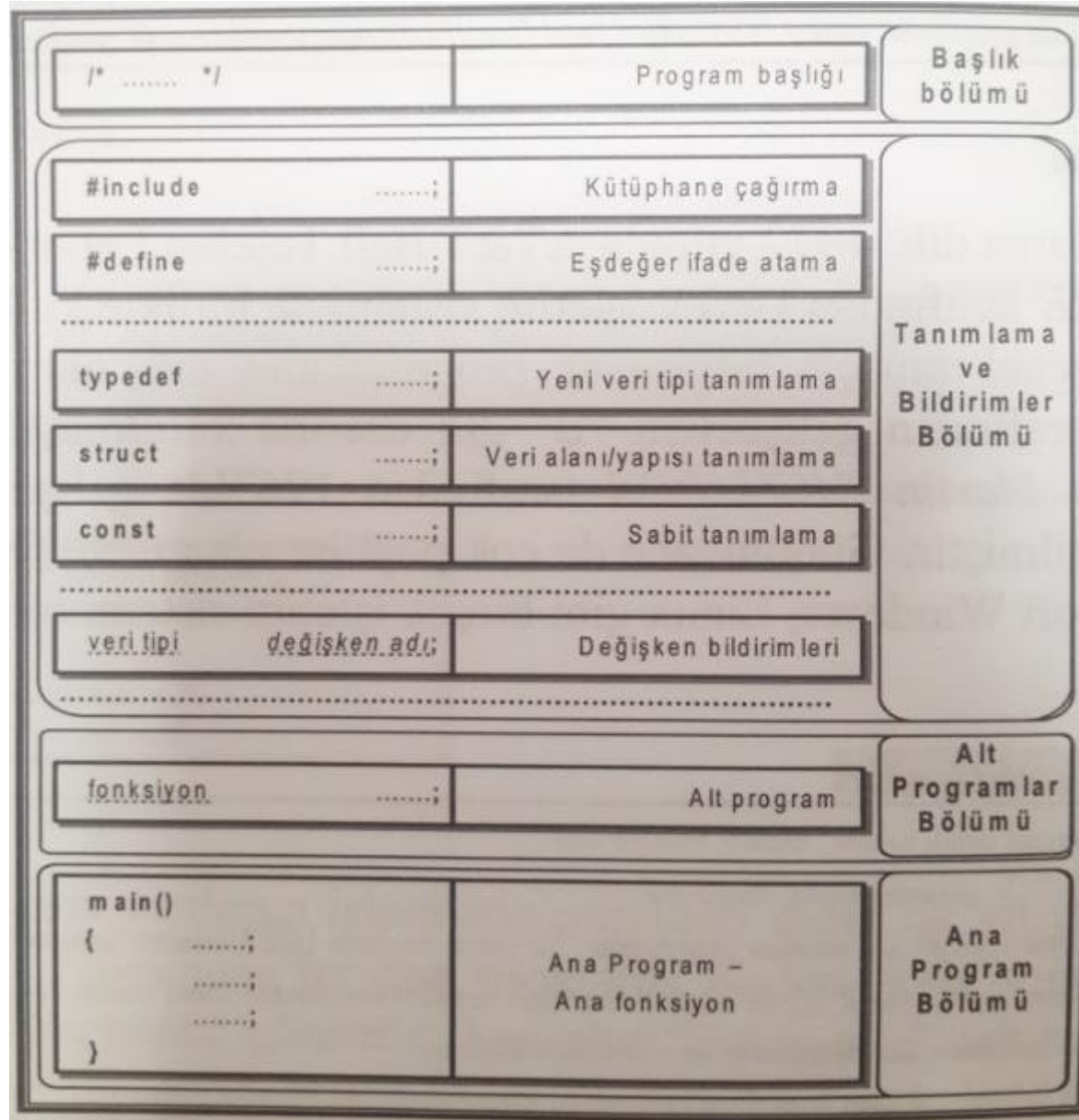
data_type variable_name;

data_type variable_name = value;

- d) Constant Definition or Initialization:** "const" is used to define constants in C programs.

const constant_name = value;

Structure of C Language



Header Part

Definition and Declaration Part

Sub Programs Part

Main Program Part

Structure of C Language

```
1  /* Program: Area Of Circle
2     Author: Alien */
3  #include<stdio.h>
4  #include<conio.h>
5
6  #define PI 3.14
7
8  void area(int);
9
10 main()
11 {
12     int radius;
13     printf("Enter Radius Of Circle ");
14     scanf("%d",&radius);
15     area(radius);
16 }
17
18 void area(int r)
19 {
20     float result;
21     result = PI*r*r;
22     printf("Area Of Circle is %f", result);
23 }
24
```

Documentation Section

Link Section

Definition Section

Global Declaration Section

Main() Function Section

Subprogram Section

C Reserved Words

Anahtar Kelimeler (Keywords)

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

Recommendations for Writing Code

- ▶ Program descriptions and document preparation should be made while programming. This is very important point that should be noted.
- ▶ Variables, constants and function names must be long enough to be selected from the meaningful words.
- ▶ If the names contains a few words, words should be separated using underscore (_) or each word should start capitalized. For example:
 - `int last_taken_bit;`
 - `void InterruptNumber();`
 - `float Mean Value = 12.7786;`
- ▶ All letters of constant should be written in capital letters. For example:
 - `#define PI = 3.14;`
 - `int STATUS 0x0379;`

Recommendations for Writing Code

- ▶ Use the TAB key to entering any sub-program part. This will increase readability. For example:

```
for(i=0; i<10; i++)  
{  
    for(j=0; j<i; j+=2)  
    {  
        do{  
            k = i + j;  
        }while(k!=0);  
    }  
}
```

Recommendations for Writing Code

- ▶ Use the space character before and after the arithmetic operators and assignment operators. This will provide a better understanding of written mathematical expression .

- ▶ For example:

```
Hmax = pow(Vo, 2) / (2 * g);
```

```
Tf = 2 * Vo / g;
```

```
Vy = Vo - g * t;
```

```
y = Vo * t - (g * t * t) / 2.0;
```

```
z = ( a * cos(x) + b * sin(x) ) * acos(y);
```

Recommendations for Writing Code

- ▶ After the program is over, review your program over and over and look for ways to better write your program.
- ▶ Try to obtain the same functions with shorter algorithms and more modularity.
- ▶ Make the necessary studies in order to understand your program.
- ▶ Transfer your knowledge and work to others in the best way.

Input/Output Library

- ▶ I/O functions are defined in standard input/output C library
 - `stdio.h`
- ▶ You need to add "stdio.h" to the beginning of the program
 - You need to do add this with the preprocessor command **#include**.

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

- ▶ Preprocessor commands begin with #.
 - `#define`

Input/Output Functions

- ▶ The I/O functions are defined in the standard input/output C library.
 - `stdio.h`
- ▶ Keyboard Input
 - `scanf` -- General formatted input
 - `getchar` -- reads a single character
- ▶ Monitor (Screen) Output
 - `printf` -- General formatted output
 - `putchar` -- writes a single character

scanf Function

- ▶ It allows data transfer to the variables entered from the keyboard.

```
scanf( "expression format", &variable list);
```

- ▶ The "*expression format*" refers to the format of data; "variable list" specifies variables to which data is to be transferred.

scanf Function

```
int number;  
  
printf("Enter one integer: ");  
scanf("%d", &number);
```

The diagram illustrates the components of the `scanf` function call. A blue bracket under the format string `"%d"` is labeled "Format part". A blue bracket under the variable address `&number` is labeled "Variable address". A red dotted arrow points from the format part to the variable address, indicating the flow of data from the input to the variable.

Format part

Variable address

scanf Examples

- ▶ “%c” char
- ▶ “%d” int
- ▶ “%f” float
- ▶ “%lf” double

```
int n;  
double d;  
char c;  
  
printf("Enter 3 values;\n");  
printf("one int, one double, and one char: ");  
scanf("%d", &n);  
scanf("%lf", &d);  
scanf("%c", &c);
```

printf Function

- ▶ Function that writes data to the screen by formatting.

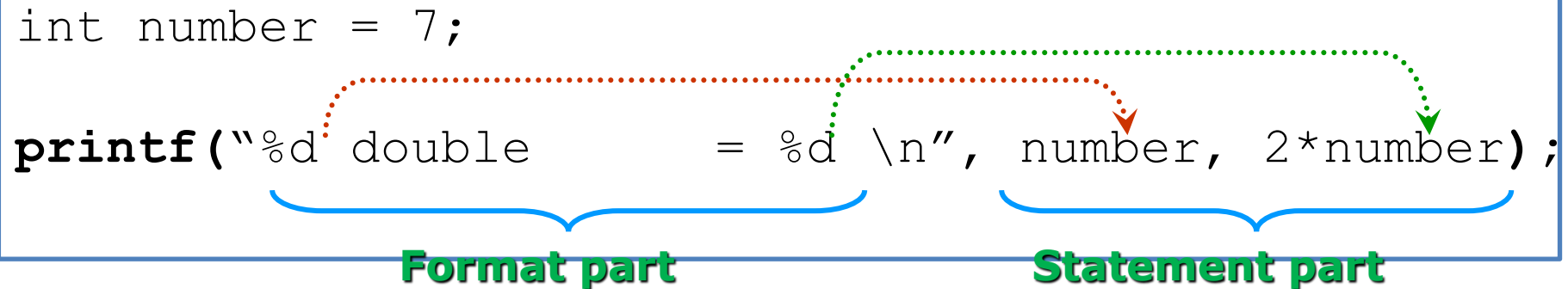
```
printf( " expression format ", variables );
```

- ▶ "expression format" generally consists of three parts.
 - Description part
 - Format part
 - Control/exit part

printf Function

```
int number = 7;
```

```
printf("%d double = %d \n", number, 2*number);
```



The diagram illustrates the components of the `printf` function call. A blue box encloses the code. A blue bracket under the string `"%d double = %d \n"` is labeled "Format part". A blue bracket under the arguments `number, 2*number` is labeled "Statement part". A red dotted line with an arrow points from the first `%d` in the format string to the variable `number`. A green dotted line with an arrow points from the second `%d` in the format string to the expression `2*number`.

Format part

Statement part

printf Function

a) Description: It is written directly to the screen in double quotes.

```
printf("Ankara");
```

b) Format: Starting with the % symbol and is part of the specified output format.

```
printf("Result: %d ", x);
```

.precision ⇒ Specifies the maximum number of characters to be displayed in.

```
printf("«Result: %.2lf ", y);
```

printf Type Setting Characters

Character	Type	Output Format
c	char	Single-byte character
hd	short	Signed decimal short int (2 byte int)
d	int	Signed decimal integer
ld	long	Signed decimal long integer
u	int	Unsigned decimal integer
x	int	Hexadecimal integer (base 16)
f	float	Signed decimal numbers
lf	double	Signed decimal numbers but much more sensitive
e	float double	Signed real numbers (scientific formatting)

printf Function

c) Control: begins with the "\" character and meaning of these signs are as follows:

Character	Mean
\a	Produce sound (alert)
\b	Move the cursor to the left (backspace)
\f	Page jump. Move beginning of the next page (formfeed)
\n	Move new line (newline)
\r	Made carriage (carriage return)
\t	Horizontal TAB
\v	Vertical TAB
\"	Write a double-quote character to the screen
\'	Write a single-quote character to the screen
\\	Write "\" character to the screen
%%	Write " % " character to the screen

printf Examples

```
double fp = 251.7366;  
int i = 25;  
printf("Real number: %.2lf \n", fp);  
printf("Right-handed integer: %10d \n", i);
```

Output:

```
Real number: 251.74  
Right-handed integer :           25
```

printf Examples

```
printf("%.5f\n", 300.0123456789);  
printf("%.14lf\n", 300.01234567890123456789);
```

300.01235

300.01234567890123

printf Examples

```
printf("%e ve %e\n",  
        300.00145678901, 0.0024) ;
```

```
3.000015e+002 ve 2.400000e-003
```

scientific view for float and double.
Note: 7 digit precision for the float.

getchar ve putchar Functions

- **getchar** takes a single character from the keyboard.
- **putchar** writes a single character to the screen.
- Example:

```
char c;

printf("Menu \n");
printf("      (a) Write C program\n");
printf("      (b) Go swimming \n");
printf("      (c) Watch TV\n");
printf("Choose one option: ");

c = getchar(); /* Take user selection */
getchar();     /* new line '\n' */
               /* put this character */
putchar('B');  /* Write B to the screen */
c = 'Z';
putchar(c);    /* Write Z to the screen */
```

References

- ▶ Doç. Dr. Fahri Vatansever, “Algoritma Geliştirme ve Programlamaya Giriş”, Seçkin Yayıncılık, 12. Baskı, 2015.
- ▶ J. G. Brookshear, “Computer Science: An Overview 10th Ed.”, Addison Wisley, 2009.
- ▶ Kaan Aslan, “A’dan Z’ye C Klavuzu 8. Basım”, Pusula Yayıncılık, 2002.
- ▶ Paul J. Deitel, “C How to Program”, Harvey Deitel.
- ▶ Bayram AKGÜL, C Programlama Ders notları