CME111 Programming Languages I

Week 8
Variable Types in C Language and Basic Input/Output Operations

Assist. 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**
- 4 byte
- from -2,147,483,648 to 2,147,483,647 (total number 4,294,967,296)

**unsigned int**
- 4 byte
- 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
double d;
d = 3.12E+5; /* 312000.0 */
d = -45.678; /* negative */
  ```

➤ Example:
  ```
  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:
  ```c
  char c;
  c = 'A';  /* Letter A */
  c = '9';  /* Number 9 */
  char c = 'c';  /* initialization */
  ```
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

<table>
<thead>
<tr>
<th>Decimal Hex Char</th>
<th>Decimal Hex Char</th>
<th>Decimal Hex Char</th>
<th>Decimal Hex Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>32</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>33</td>
<td>21</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>34</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>35</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>36</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>37</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>38</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>39</td>
<td>27</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>41</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>42</td>
<td>2A</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>43</td>
<td>2B</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>44</td>
<td>2C</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>45</td>
<td>2D</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>46</td>
<td>2E</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>47</td>
<td>2F</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>48</td>
<td>30</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>49</td>
<td>31</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>51</td>
<td>33</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>52</td>
<td>34</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>53</td>
<td>35</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>54</td>
<td>36</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>55</td>
<td>37</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>56</td>
<td>38</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>57</td>
<td>39</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>58</td>
<td>3A</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>59</td>
<td>3B</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>60</td>
<td>3C</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>61</td>
<td>3D</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>62</td>
<td>3E</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>63</td>
<td>3F</td>
</tr>
</tbody>
</table>

**Note:** The ASCII characters include digits, letters, punctuation marks, and control characters such as space, newline, and backspace.
Special Characters

Characters are shown in single quotes

– How we will show quotation marks?

```c
char c;
c = ''; // Incorrect
```

```c
char c;
c = '\'; // Correct
```

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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1 byte</td>
<td>-128 : 127</td>
</tr>
<tr>
<td>unsigned char</td>
<td>1 byte</td>
<td>0 : 255</td>
</tr>
<tr>
<td>short</td>
<td>2 byte</td>
<td>-32768 : 32767</td>
</tr>
<tr>
<td>unsigned short</td>
<td>2 byte</td>
<td>0 : 65535</td>
</tr>
<tr>
<td>int</td>
<td>4 byte</td>
<td>-2147483648 : 2147483647</td>
</tr>
<tr>
<td>unsigned int</td>
<td>4 byte</td>
<td>0 : 4294967295</td>
</tr>
<tr>
<td>float (7 precision)</td>
<td>4 byte</td>
<td>1.175494e-38 : 3.402823e+38</td>
</tr>
<tr>
<td>double (16 precision)</td>
<td>8 byte</td>
<td>2.225074e-308 : 1.797693e+308</td>
</tr>
</tbody>
</table>
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:* standard 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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/* ... */</code></td>
<td>Program başlığı</td>
</tr>
<tr>
<td><code>#include</code></td>
<td>Kütüphane çağırma</td>
</tr>
<tr>
<td><code>#define</code></td>
<td>Eşdeğer ifade atama</td>
</tr>
<tr>
<td><code>typedef</code></td>
<td>Yeni veri tipi tanımlama</td>
</tr>
<tr>
<td><code>struct</code></td>
<td>Veri alanı/yapısı tanımlama</td>
</tr>
<tr>
<td><code>const</code></td>
<td>Sabit tanımlama</td>
</tr>
</tbody>
</table>

## Definition and Declaration Part

<table>
<thead>
<tr>
<th>Type</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>veri tipi</code></td>
<td>değişken adı;</td>
</tr>
<tr>
<td></td>
<td>Değişken bildirimleri</td>
</tr>
</tbody>
</table>

## Sub Programs Part

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fonksiyon</code></td>
<td>Alt program</td>
</tr>
</tbody>
</table>

## Main Program Part

```c
main()
{
    ........;
    ........;
    ........;

Ana Program – Ana fonksiyon
```
Structure of C Language

/* Program: Area Of Circle
   Author: Alien */

#include<stdio.h>
#include<conio.h>

#define PI 3.14

void area(int);

void area(int r)
{
    float result;
    result = PI*r*r;
    printf("Area Of Circle is %f", result);
}

main()
{
    int radius;
    printf("Enter Radius Of Circle ");
    scanf("%d", &radius);
    area(radius);
}
# C Reserved Words

<table>
<thead>
<tr>
<th>Anahtar Kelimeler (Keywords)</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
</tr>
<tr>
<td>break</td>
</tr>
<tr>
<td>case</td>
</tr>
<tr>
<td>char</td>
</tr>
<tr>
<td>const</td>
</tr>
<tr>
<td>continue</td>
</tr>
<tr>
<td>default</td>
</tr>
<tr>
<td>do</td>
</tr>
</tbody>
</table>
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:

```c
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:

\[
\begin{align*}
H_{\text{max}} &= \text{pow}(V_0, 2) / (2 * g); \\
T_f &= 2 * V_0 / g; \\
V_y &= V_0 - g * t; \\
y &= V_0 * t - (g * t * t)/ 2.0; \\
z &= ( a * \cos(x) + b * \sin(x) ) * \acos(y);
\end{align*}
\]
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`.

- Preprocessor commands begin with #.
  - `#define`
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
It allows data transfer to the variables entered from the keyboard.

```c
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.
int number;

printf("Enter one integer: ");

scanf("%d", &number);
scanf Examples

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

```c
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.

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

"expression format" generally consists of three parts.
- Description part
- Format part
- Control/exit part
int number = 7;

printf(“%d double = %d \n”, number, 2*number);
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

<table>
<thead>
<tr>
<th>Character</th>
<th>Type</th>
<th>Output Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>char</td>
<td>Single-byte character</td>
</tr>
<tr>
<td>hd</td>
<td>short</td>
<td>Signed decimal short int (2 byte int)</td>
</tr>
<tr>
<td>d</td>
<td>int</td>
<td>Signed decimal integer</td>
</tr>
<tr>
<td>ld</td>
<td>long</td>
<td>Signed decimal long integer</td>
</tr>
<tr>
<td>u</td>
<td>int</td>
<td>Unsigned decimal integer</td>
</tr>
<tr>
<td>x</td>
<td>int</td>
<td>Hexadecimal integer (base 16)</td>
</tr>
<tr>
<td>f</td>
<td>float</td>
<td>Signed decimal numbers</td>
</tr>
<tr>
<td>lf</td>
<td>double</td>
<td>Signed decimal numbers but much more sensitive</td>
</tr>
<tr>
<td>e</td>
<td>float double</td>
<td>Signed real numbers (scientific formatting)</td>
</tr>
</tbody>
</table>
### printf Function

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>\a</td>
<td>Produce sound (alert)</td>
</tr>
<tr>
<td>\b</td>
<td>Move the cursor to the left (backspace)</td>
</tr>
<tr>
<td>\f</td>
<td>Page jump. Move beginning of the next page (formfeed)</td>
</tr>
<tr>
<td>\n</td>
<td>Move new line (newline)</td>
</tr>
<tr>
<td>\r</td>
<td>Made carriage (carriage return)</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal TAB</td>
</tr>
<tr>
<td>\v</td>
<td>Vertical TAB</td>
</tr>
<tr>
<td>&quot;</td>
<td>Write a double-quote character to the screen</td>
</tr>
<tr>
<td>'</td>
<td>Write a single-quote character to the screen</td>
</tr>
<tr>
<td>\</td>
<td>Write &quot;\&quot; character to the screen</td>
</tr>
<tr>
<td>%%%</td>
<td>Write &quot; % &quot; character to the screen</td>
</tr>
</tbody>
</table>
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

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

300.01235
300.01234567890123
printf Examples

```c
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** and **putchar** Functions

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

```c
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'*/
  putchar('B');  /* Write B to the screen */
  c = 'Z';
  putchar(c);    /* Write Z to the screen */
```
References


► Paul J. Deitel, “C How to Program”, Harvey Deitel.

► Bayram AKGÜL, C Programlama Ders notları