

Chapter
2

DATATYPES IN C

*Get on the
Fast Track!*



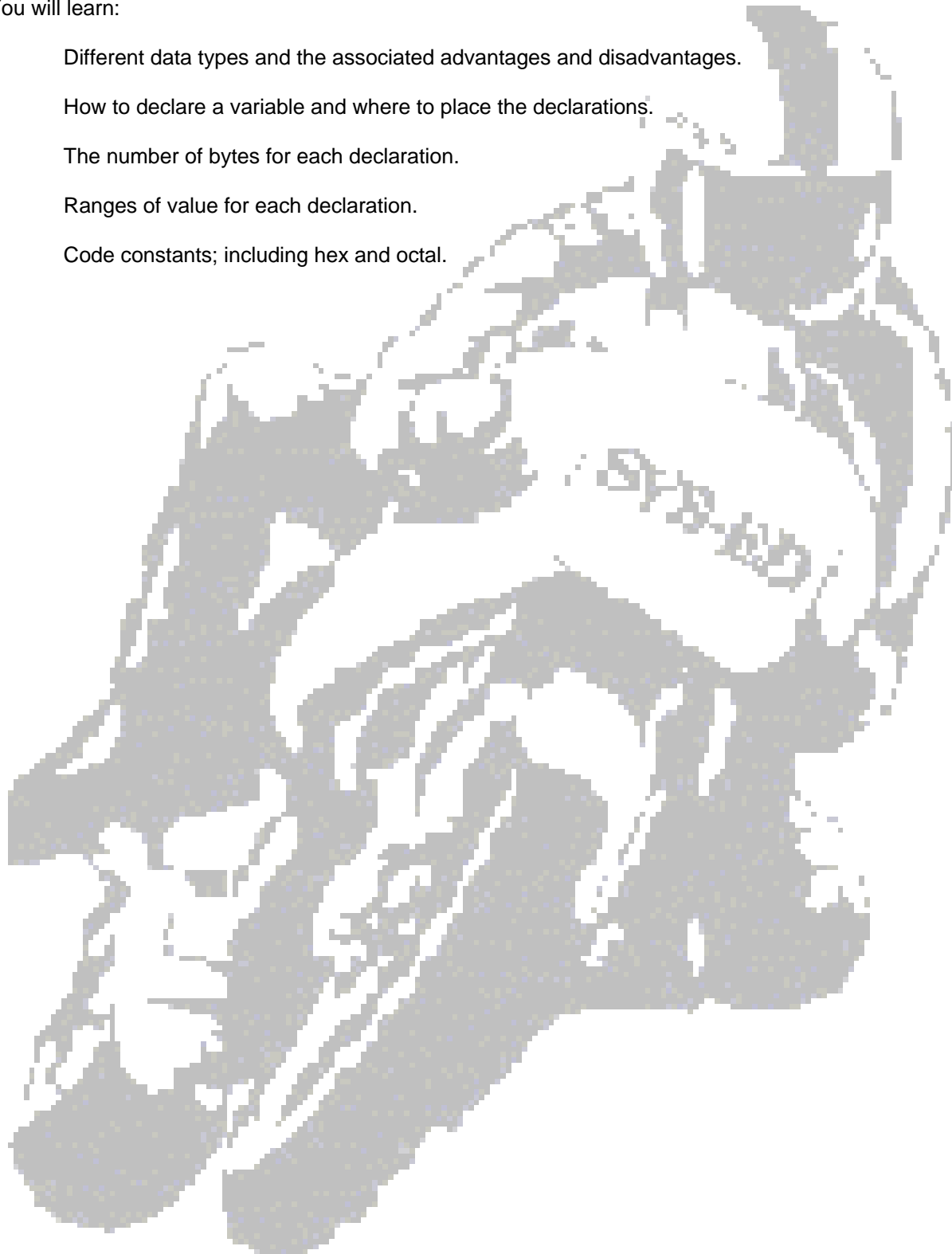
TM

**SYS-ED/
COMPUTER
EDUCATION
TECHNIQUES, INC.**

Objectives

You will learn:

- C Different data types and the associated advantages and disadvantages.
- C How to declare a variable and where to place the declarations.
- C The number of bytes for each declaration.
- C Ranges of value for each declaration.
- C Code constants; including hex and octal.



1 Datatypes, Variables, and Constants

Datatypes

There are two fundamentally different types of numeric data values:

- C integer
- C floating point

From these we generate two other primary data types:

- C character
- C double precision

C uses the usual simple data types:

- C characters (char)
- C integers (int)
- C numbers with fractional components (float and double)

C allows variants of simple data types which can vary not only information but can contain addresses (storage locations) often called pointers.

Void is a new keyword which was recently added by ANSI for the C standard. Void may either declare a function as returning no value or may be used to create generic pointers.

2 C Character Types (char)

A character is a one-byte space in memory.

The type name for character is char. Character data sets may either be constants or variables. Character variables are declared using the keyword "char" at the beginning of the programming block.

```
char sex, party;
```

Character constants are written as letters or symbols inside single quotation marks:

```
char sex, party;  
.  
.  
sex = 'f';  
party = 'd';
```

When we assign particular values to our character variables, the compiler will store these values in the reserved memory locations.

C allocates one byte for storing a character giving a maximum of 256 characters values.

Character variables are interpreted as values in the range of -128 to 127.

Negative char don't have a useful interpretation, but values can be used in operations to indicate an error or end of input.

To assign a value to a char variable, put the value between single quotation marks, and use the assignment operator.

3 C Integer Types (int)

An integer is a whole number such as 1,2 or 3 or -1 or 0, that doesn't contain fractional parts. An integer may have a positive or a negative value.

Signed Integers

- C An unsigned integer is represented as a positive number or zero in C. A signed integer is necessary for a negative number.
- C Integer constants are written as they are in standard arithmetic. Zero is 0. The number one is written as 1 or +1. An unsigned number defaults to a positive. Minus one is -1, a negative number must always be preceded by a negative sign.

4 Variables

A variable is a space in the computer's memory set aside for certain kinds of data and given a name for easy reference.

```
void main(void)
{
    int num;
    num = 2;
    printf("This is the number two: %d", num);
}
```

5 Integer Variable

When you know in advance that the integer will always be positive, such as used in counting, you can declare the variable to be unsigned. You can use the keyword `unsigned` or `unsigned int` to designate this variable as a positive integer.

```
unsigned int rank;  
  
unsigned rank;
```

Declaring an integer to be unsigned will usually double the size of its largest positive value, depending on the compiler, since all possible negative values will be eliminated.

```
int num;
```

The above is an example of variable declaration. In a C program all variables must be declared. If you have more than one variable of the same type, you can declare them all with one type name, separating the variable names with commas.

```
int apples, oranges, pears;
```

All variables must be declared to specify their name and type.

```
1  /* c2_p6.c          */
2  /* int vs unsigned */
3  #include <stdio.h>
4  void main(void)
5  {
6      int      fld1;
7      unsigned fld2;
8
9      fld1 = 2147483647;    /* for 16 bit systems use      */
10     fld2 = 2147483647;    /* fld1 = 32767; & fld2 = 32767; */
11
12     printf("The values of fld1 and fld2 are: %d %u",
13           fld1,fld2);
14
15     fld1 = fld1 + 1;
16     fld2 = fld2 + 1;
17
18     printf("\nThe values of fld1 and fld2 are: %d %u",
19           fld1,fld2);
20
21     /* Sample Output
22
23     for 32 bit systems:
24
25     The values of fld1 and fld2 are: 2147483647 2147483647
26     The values of fld1 and fld2 are: -2147483648 2147483648
27
28
29     for 16 bit systems:
30
31     The values of fld1 and fld2 are: 32767 32767
32     The values of fld1 and fld2 are: -32768 32768
33     */
34
35 } /* End of main */
```

Most variables in C always occupy the same amount of memory space, but the number of bytes occupied by type `int` changes, depending on the computer being used.

2 bytes (16 bits) 8088, 8086, 80286	4 bytes (32 bits) 80386
--	----------------------------

The number of bits used corresponds to the data path of the computer; it's the size that can be read and written most efficiently.

The value of two-byte integers can vary from -32,768 to 32,767, while four-byte integers can vary from -2,147,483,648 to 2,147,483,647.

If you want to guarantee a two-byte variable, regardless of the type of machine, use type `short`, which means short integer. If you want to guarantee a four-byte integer use type `long`.

6 Variable Types

Character Variable

A character variable is a one-byte space in memory. The type name for a character is char.

```
1  /* c2_p8.c      */
2  /* Use of char */
3  #include <stdio.h>
4  void main()
5  {
6      char  charGender;
7
8      charGender = 'F';
9          /* note it is an upper F */
10
11     printf("It can be used as a number %d\n", charGender);
12     printf("It can be used as a single character %c\n",charGender);
13
14     /* It can be used in math expressions */
15     charGender = charGender + 1;
16     printf("\nIt can be used as a number %d\n", charGender);
17
18     /* It can be used in comparing a character */
19     if (charGender == 'G')
20         printf("charGender contains a G \n");
21 }/* End of main */
22
23 /* sample Output
24 It can be used as a number 70
25 It can be used as a single character F
26
27 It can be used as a number 71
28 charGender contains a G
29 */
```

7 Declaring Integer Variables

Integer variables must be declared before they are used. The declaration is placed at the top of a block of code.

```
1  {
2  int a;
3  .
4  }
5
6  {
7  int a,b,c;
8  .
9  .
10 .
11 }
12
13 {
14 int graduating_class, rank, student_number;
15 .
16 .
17 .
18 }
```

Once a variable has been declared an int, it can be assigned an integer value by the assignment operator =.

For example:

```
1  int graduating_class, rank, student_number;
2  .
3  .
4  .
5  student_number = 4221;
6  rank = 519;
7  graduating_class = 1990;
```

Here the "=" symbolizes the operation of assigning constants to variable names, it is not a statement of equality or a mathematical operation. $x = x + 1$ in this instance is quite legal, it simply increments the value of x by one.

Variables can be declared in any order. The order of declaration need not match the order in which they appear in a block of code. Variables are usually placed in an organized order based upon their function.

Integer constants have a fixed value and are never declared. There may be times when you need to use a constant that's larger than an int. In such a case, you can make it a long integer. If a constant's value is too large to be stored in the space normally allotted for an integer, the compiler will automatically treat it as a long constant.

Integer Constants

1L	long integer constant
1 ul or 1 UL	unsigned long

8 Long and Short Integers

Two other versions of the int type, short and long, are requests for different amounts of storage allocations, depending on the implementation.

Long integer variables must be declared using the keyword "long":

```
long int head_count;
```

The use of long integers requires a little more space in memory and the program may run a bit slower, but the range of values you can use is expanded enormously.

C allows the abbreviation of short for short int and long for long int.

You can define variable of types unsigned short int, unsigned long int, signed short int and signed long int.

The same amount of storage will be allocated to int and unsigned int variables.

The storage for a short int will be less than or equal to the storage allocation for an int.

The storage for a long int will be greater than or equal to the storage allocation of an int.

C also requires char variable to be stored in one byte and insists that a double variable be allocated at least as much space as a float.

Types whose storage sizes are most likely to differ from system to system include short int, int, and long int. Such differences may affect the portability of your programs.

9 Integers

Short int is 2 and long int is 4 bytes.

Integer type is used to represent whole numbers within a specified range of values. Int variables are stored in a 16-bit area of memory on PC's. This allows the storage of 65,536 (2^{16}).

Values are represented as -32,768 to 32,767, or (-2^{15} to $2^{15} - 1$). One bit represents the numbers sign positive or negative.

In larger systems int variables may be allocated to 32 bits, this allows for a possible value range from (-2^{31} to $2^{31} - 1$).

If you use a integer larger than what the system is capable of storing you will get unreliable results. Because of the way the integer values are represented, the system ends up reading a very large number as a very large negative number, due to the overflow in the representation of the integer.

Commas are not allowed when writing a number in C.

Beware of overflow when doing computations using integers.

10 The Float Type

The most usual type of floating-point, type float, occupies four bytes or 32 bits and can hold numbers from 10^{-38} to 10^{+38} , with between six and seven digits for precision.

Floating-point data values are approximations of real numbers, usually expressed as decimal fractions. Real numbers that are very large or very small are often written in exponential or scientific notation. A real number is a number which may have a fractional part. An integer is a whole number. Either the decimal fraction or the exponential style is acceptable for writing floating-point constants in C. Because we can't do superscripts on most terminals, the exponent is denoted by an e or E preceding the exponential value.

The keyword to declare a floating-point variable is float.

These numbers are referred to as floating point numbers because the value is represented in three parts.

- C A sign
- C Numerical value (the fractional part of which is referred to as the "mantissa")
- C Exponent to indicate where the decimal point is placed

10.1 Float

The number 250.450 can be represented in the following ways:

250.450E0	25.0450E1	2.50450E2	2505.50E-1	25045.0E-2
-----------	-----------	-----------	------------	------------

E0	moves decimal zero places.
E1	moves decimal one place to the right.
E2	moves decimal two places to the right.
E-1	moves decimal one place to the left.
E-2	moves decimal two places to the left.

- C When representing this number the exponent makes the decimal point float.
- C C allocates 32 bits to variables of type float.
- C C provides a range of 10^{-38} to 10^{+38} . Because of the sign bit you can allocate the same number of positive and negative values.
- C Because numerical values are represented in binary format, rounding errors can occur in numerical calculations. These errors can become significant, particularly if many calculations are performed.

10.2 Float Example

```
1  /* c2_p15.c */
2  /* Float Example */
3  #include <stdio.h>
4  void main()
5  {
6      float fSalary;
7      int   iSalary;
8      /* The following two statements will generate warning messages */
9      fSalary = 250.75;
10     iSalary = 250.75; /* note the decimal on an integer */
11     printf("Floating salary is %f\n",fSalary);
12     printf("Integer salary is %d\n",iSalary);
13
14 } /* End of main */
15
16 /* sample Output
17 Floating salary is 250.750000
18 Integer salary is 250
19 */
```

11 The Double Type

Double type is for double precision floating point numbers. All double values are stored in a similar manner to float values as signs, mantissas (fractional parts of a number), and exponents. To minimize the problem of approximation found in floating-point decimal, double-precision data is used for accuracy, although some speed and compactness of program size is lost.

64 bits of storage are allocated for the double-precision value, twice the storage allocated for the floating-point value.

Values of the double-precision (or double type) are more precise than the floating-point variable and eliminate much of the inaccuracy of arithmetic and rounding errors.

Information is stored in the same manner as the float type, but the range and number of possible values is much greater and allows for more precision. Values of the double variable have 17 significant digits vs. 11 significant digits for the float. All real-number constants are handled as double-precision values by the compiler.

Double-precision variables are declared using the keyword `double`.

A double allows for the representation of values ranging from 10^{-308} to 10^{+308} , with about 15 digits for precision.

There is an equal range for both positive and negative numbers.

The main difference between the float and the double is the amount of space allocated from numbers of these types. Because of the way float and double values are represented, not all feasible values within a permissible range can be represented. Intermediate computations in an expression may introduce rounding errors that could slightly change the value of a particular variable. Such values can therefore be only approximately correct or nearly equal. For most calculations, you need not be concerned about the effect of such errors, just be aware they can occur.

12 Alternate Number Systems

C will accept integer constants written in either decimal, octal or hexadecimal number systems.

Octal values must always be preceded by a zero.

```
07
0377
07777
02000000
```

You should never write a decimal constant that begins with a zero.

Hex values must always be preceded by a zero and either a small x or upper-case X.

```
0x7c
0x1a
0X7fff
0X12345
```

It is possible to use the L suffix to specify long octal or hex constants. For the compiler there is no difference between integer constants written in decimal notation, octal or hex; the compiler stores them all as binary.

```
1  /* c2-p19.c  */
2  /* Alternate Number System Constants */
3  #include <stdio.h>
4  void main()
5  {
6      int  fld;
7      fld = 10;
8      printf("fld is %d\n",fld);
9      fld = 010;
10     printf("fld is %d\n",fld);
11     fld = 0x10;
12     printf("fld is %d\n",fld);
13
14 } /* End of main */
15 /* Sample Output
16 fld is 10
17 fld is 8
18 fld is 16
19 */
```

13 Special Constants

For those characters not defined by an escape sequence, it is acceptable to use integer constants to define them. If we wanted to use the ASCII code which defines the bell character of 7 to make the terminal beep, we need only identify the variable name `bell` as a bell character in our programs.

```
char bell;  
.  
.  
bell = 7;  
putchar (bell);
```

Whenever the expression `bell` is used in the program it will cause the terminal to beep.

It is better to use numerical escape sequences written in the 'ddd' or octal value from the ASCII table. The preferred way of setting the character variable `bell` to the ASCII control code BEL is:

```
char bell  
.  
.  
bell = '007';
```

('07' or '\7' will also ring bells)

ASCII control codes may also be written using the octal or hexadecimal code. The only limitation is that the leading zero that defines the number as octal may only be used with two digit codes (not three). For an escape code, you may use up to three digits and the number is assumed to be in octal form.

14 Constants

Integer Constants

Number is written without commas. Numbers can be written in octal, decimal or hexadecimal form.

Character Constants

Character constants are written within single quotes. Character constants using the alphabetic letter or using the ASCII code assigned to it in octal, decimal or hexadecimal.

String Constants

String constants consist of a sequence of characters. The string constant must be enclosed between two double quotes, or it will not be identified as a string constant.

```
"this is a string constant.\n"
'this is NOT a string constant.\n'
```

Floating-point Value and Double-precision Value Constants

When writing a constant, the only difference between a float and a double is in the feasible range of values. If you try to assign a number too large for a floating-point variable, you will get a floating point overflow error message. The value must have a number portion; it may be preceded by a sign and may be followed by an exponent portion.

The number portion has one of the following forms:

.54	Decimal point followed by one or more digits.
54.	One or more digits followed by a decimal point.
54.5	One or more digits followed by a decimal point, followed by one or more digits.

C requires a decimal point or exponent portion for floating-point values. If there is no decimal point or exponent portion, C identifies the value as an integer (int), no matter what your intention.

Character type and the integer types also have unsigned version; unsigned char, unsigned short, unsigned int, and unsigned long, which change the range of numbers the type can hold. The unsigned int type holds numbers from 0 to 65,535 rather than from -32,768 to 32,767 as the regular int type does.

There are no string variable types in C. Instead strings are represented by arrays of characters.



15 Initializing Variables

Here we declare and give a value to a variable.

```
1 double logE = 2.718281828459;
2 float x = 12.345;
3 int rank = 1120;
4 char sex = 'f';
```

It's possible to combine a variable declaration with an assignment operator so that a variable is given a value at the same time it's declared.

```
1 void main(void)
2 {
3     int event = 5;
4     char heat = 'C';
5     float time = 27.25;
6
7     printf("The winning time in heat %c ", heat);
8     printf("of event %d was %f.", event, time);
9 }
```