Programming for Engineers

Lecture 2B: Data Types

Course ID: EE057IU

Data Types

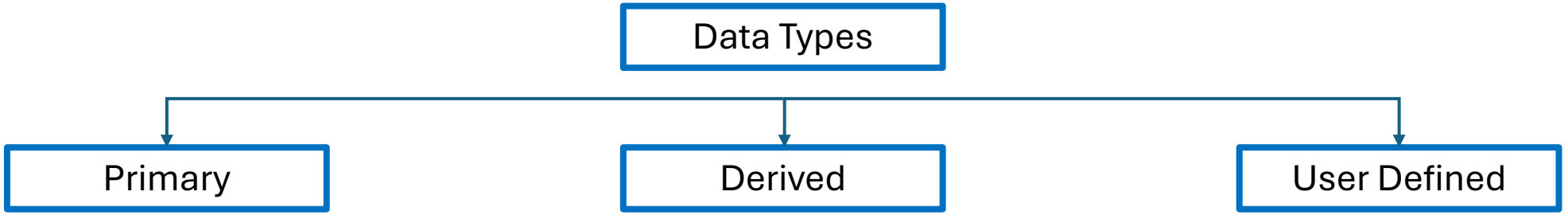
- In the C Programming language, data types refer to a broad system used for declaring variables or function of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern is interpreted

→ **Primitive:** are most basic data types that are used for representing simple values such as integers, float, characters, etc.

→ **User Defined:** are defined by the user yourself

→ **Derived:** derived from the primitive or build-in datatypes

Types of Data Types



- → Integer
- → Character
- → Floating
- → Double
- → Void

- → Array
- → Pointer
- → Function
- → Reference

- → Structure
- → Enumerated
- → Typedef
- → Class
- → Union

Primitive Data Types

Data Type	Description	Size (byte)	Range (Signed)	Range (unsigned)	Format
char	Character type	1	-128 to 127	0 to 255	%c
short	Short integer	2	-32,768 to 32,767	0 to 65,535	%hd
int	Standard integer	4	-2^31 to 2^31 - 1	0 to 2^32 - 1	%d
long	Long integer	4 or 8	-2^63 to 2^63 - 1	0 to 2^64 - 1	%ld
long long	Long long integer	8	-2^63 to 2^63 - 1	0 to 2^64 - 1	%lld
float	Single-precision floating	4	±3.4E-38 to ±3.4E+38		%f
double	Double-precision floating	8	±1.7E-308 to ±1.7E+308		%lf
long double	Extended-precision floating	12 or 16	Greater than ‘double’		%Lf
_Bool	Boolean type	1	0 (false) or 1(true)		%d

CHAR Data Type

- Allow its variable to store only a single character
- **Range**: (-128 to 127) or (0 to 255)
- **Size**: 1 byte
- **Format Specifier**: %c (character)
  or %d (integer in ASCII)
- **Syntax**: **char** var_name;

Character	ASCII Code
' '	32
'*'	42
'A'	65
'B'	66
'z'	90
'a'	97
'b'	98
'z'	122
'0'	48
'9'	57

CHAR and ASCII

```
#include <stdio.h>

int main()
{
    char myChar = ‘A’;
    printf(“The ASCII form of %c is %d\n”, myChar, myChar)
    return 0;
}
```

Output:

The ASCII form of A is 65

```
#include <stdio.h>

int main(){
    char a = 'a';
    char c;
    printf("Value of a: %c\n", a);
    a++;
    printf("Value of a after increment is: %c\n", a);
    c = 99;
    printf("Value of c: %c", c);
    return 0;
}
```

Output:

Value of a: a

Value of a after increment is: b

Value of c: c

INT Data Type

- The integer datatype in C is used to store the integer numbers
- **Range**: -2,147,483,648 to 2,147,483,647
- **Size**: 4 byte
- **Format Specifier**: %d
- **Syntax**: **int** var_name;

INT Data Type

- Example 1: Basic Integer Types and Their Format

```c
#include <stdio.h>

int main()
{
    // Integer value with positive data.
    int a = 42;

    // Integer value with negative data.
    int b = -42;

    // Unsigned integer value.
    unsigned int c = 300U;

    // Long integer value.
    long int d = 123456789L;

    // Long long integer value.
    long long int e = 987654321012345LL;

    printf("Integer value with positive data: %d\n", a);
    printf("Integer value with negative data: %d\n", b);
    printf("Unsigned integer value: %u\n", c);
    printf("Long integer value: %ld\n", d);
    printf("Long long integer value: %lld\n", e);

    return 0;
}
```

Output

```vbnet
Integer value with positive data: 42
Integer value with negative data: -42
Unsigned integer value: 300
Long integer value: 123456789
Long long integer value: 987654321012345
```

INT Data Type

- Example 2: Demonstrating Type Modifiers and Ranges

```c
#include <stdio.h>

int main()
{
    // Basic integer value.
    int basic = 100;

    // Long integer value.
    long longVal = 100000L;

    // Unsigned long integer value.
    unsigned long ulongVal = 100000UL;

    // Long long integer value.
    long long llongVal = 10000000000LL;

    // Unsigned long long integer value.
    unsigned long long ullongVal = 10000000000ULL;

    printf("Basic integer value: %d\n", basic);
    printf("Long integer value: %ld\n", longVal);
    printf("Unsigned long integer value: %lu\n", ulongVal);
    printf("Long long integer value: %lld\n", llongVal);
    printf("Unsigned long long integer value: %llu\n", ullongVal);

    return 0;
}
```

Output

```vbnet
Basic integer value: 100
Long integer value: 100000
Unsigned long integer value: 100000
Long long integer value: 10000000000
Unsigned long long integer value: 10000000000
```

FLOAT Data Type

- The float datatype in C is used to store the floating-point values. Float in C is used to store both decimal and exponential values
- **Range**: 1.2E-38 to 3.4E+38
- **Size**: 4 byte
- **Format Specifier**: %f
- **Syntax**: **float** var_name;

DOUBLE Data Type

- The float datatype in C is used to store the floating-point values with double precision.
- **Range**: 1.7E-308 to 1.7E+308
- **Size**: 8 byte
- **Format Specifier**: %lf
- **Syntax**: **double** var_name;

FLOAT and DOUBLE Data Type

- Basic usage of ‘float’ and ‘double’

```c
#include <stdio.h>

int main() {
    // Float and double variable declarations and initializations
    float f = 3.14f;           // Single-precision floating-point
    double d = 3.14159265358979; // Double-precision floating-point

    // Printing float and double values
    printf("Float value: %f\n", f);     // %f for float (default is 6 decimal places)
    printf("Double value: %lf\n", d);   // %lf for double (default is 15-16 decimal places

    return 0;
}
```

Output

```kotlin
Float value: 3.140000
Double value: 3.141593
```

FLOAT and DOUBLE Data Type

- **Precision and Formatting**

1. **Float**
   - Size: 4 bytes
   - Precision 6-7 decimal places
2. **Double**
   - Size: 8 bytes
   - Precision 15-16 decimal places

c Copy code

```c
#include <stdio.h>

int main() {
    // Float and double variable declarations
    float f = 1.234567f;
    double d = 1.234567890123456;

    // Printing with specified precision
    printf("Float value with 2 decimal places: %.2f\n", f);
    printf("Float value with 5 decimal places: %.5f\n", f);
    printf("Double value with 10 decimal places: %.10lf\n", d);

    return 0;
}
```

Output of Example 2:

sql Copy code

```sql
Float value with 2 decimal places: 1.23
Float value with 5 decimal places: 1.23457
Double value with 10 decimal places: 1.2345678901
```

Data type conversions

- Grade average example
- $class\ average = \frac{\sum grade}{number\ of\ students}$
- Grade and number of students can be integers
- Averages do not always evaluate to integers values, needs to be floating point for accuracy

Explicit conversions

- Dividing two integers results in integer division in which any fractional part of the calculation is truncated
- To produce a floating-point calculation with integer values, we create temporary values that are floating-point numbers.
- C provides the unary cast operator to accomplish this task
- $average = (float)total/counter;$
- Includes the cast operator $(float)$, which creates a temporary floating-point copy of its operand, $total$.
- Using a cast operator in this manner is called explicit conversion.

Explicit conversions

Example: Write an program to find the average of two integers