Type Casting in C

Course: Introduction to Programming and Data Structures

Dr. Laltu Sardar

Institute for Advancing Intelligence (IAI),
TCG Centres for Research and Education in Science and Technology (TCG Crest)

tcg crest
Inventing Harmonious Future

1. Type Casting in C
   - What is Typecasting?
   - Types of Typecasting
   - Examples of Implicit Typecasting
   - Examples of Explicit Typecasting
   - Some special typecastings
   - Importance of Typecasting
   - Common Use Cases

Type Casting in C

What is Typecasting?

- Typecasting is the process of converting one data type into another.
- It allows the programmer to explicitly specify how a value should be treated by the compiler.
- Typecasting can be used to perform operations on variables of different types.
- Can occur/may require during **Assignment** and **Expression**

Types of Typecasting

- **Implicit Typecasting (Automatic Conversion):**
  - `smaller type` $\rightarrowtail$ `larger type`
  - Performed automatically by the compiler.
  - In binary operator – **Lower type promoted to Higher type**
  - Example: `int` to `float`.
- **Explicit Typecasting (Manual Conversion):**
  - `larger type` $\rightarrowtail$ `smaller type`
  - Performed manually by the programmer using cast operator.
  - Used to convert a larger data type to a smaller data type.
  - Example: `float` to `int`.

Examples of **Implicit** Typecasting

```c
int main() {
    int a = 10;         // Integer type
    float b = 5.5;      // Float type

    //1. Implicit type conversion during arithmetic operation
    float result = a + b;  // 'a' is implicitly converted to float

    printf("Result of adding int and float: %.2f\n", result);

    //2. Imp. type conv. when assigning a smaller to a larger type
    double larger_result = result;  // 'result' (float) is implicitly converted to double

    printf("Value after Imp. conv. to double: %.2lf\n", larger_result);

    return 0;
}
```

Output:

```
Result of adding int and float: 15.50
Value after Imp. Conv. to double: 15.50
```

Examples of **Explicit** Typecasting

```c
int main() { //Example:  Explicit typecasting
    int intVar = 10;  // Integer type
    double doubleVar = 5.75; //double type

    // Example 1: Casting from double to int
    // (double to int) results in loss of decimal part
    int castedInt = (int) doubleVar;
    printf("Original double value: %f\n", doubleVar);
    printf("After casting to int: %d\n", castedInt);

    // Example 2: Casting from int to double
    //intVar converted to double to retain precision in division
    double result = (double) intVar / 3;
    printf("Result of division after casting int to double: %f\n",
         result);
    // Example 3: Casting characters to integer
    char charVar = 'A';
    printf("Character: %c, ASCII value after casting: %d\n",
        charVar, (int)charVar);
    return 0;
}
```

Some special typecasting

```c
int main() {
    // 1. Typecasting that does not make sense
    double pi = 3.14159;
    // Casting the address of a double to int directly doesn’t make sense, because types are incompatible and can cause undefined behavior.
    int invalidCast = (int) &pi; // Does not make sense
    printf("Invalid cast of address of pi to int: %d (Address in int form)\n", invalidCast);

    // 2. Typecasting that is not automatic (explicit cast required)
    int intVar = 5;
    double result;
    // Without explicit cast, integer division occurs, losing precision.
    result = (double)intVar / 2; // Explicit cast to double is required here
    printf("Result of casting int to double: %.2f\n", result);

    return 0;
}
```

Some special typecasting

```c
int main() {
    // 3. Complex typecasting (Casting between pointer types)
    int arr[5] = {1, 2, 3, 4, 5};
    void *voidPtr = (void *)arr;   // Storing int array as void pointer
    int *intPtr = (int *)voidPtr;  // Explicitly casting back to int pointer
    printf("Complex cast: First element of array through void pointer: %d\n", intPtr[0]);

    // 4. Be cautious: Signed vs Unsigned casting
    int signedVar = -10;
    unsigned int unsignedVar = (unsigned int)signedVar;
    // Casting signed to unsigned: bit pattern remains the same
    printf("Be cautious! Signed to Unsigned cast: %u\n", unsignedVar);

    return 0;
}
```

Importance of Typecasting

- Ensures that operations are performed with the correct data type.
- Helps in preventing data loss when converting between types.
- Allows for the manipulation of different data types in expressions.
- Useful in situations where a specific data type is required.

Common Use Cases of Typecasting

- Converting a `float` to an `int` when precision is not needed.
- Casting a `char` to an `int` to get its ASCII value.
- Converting data types when interacting with functions that require specific types.
- Converting pointers from one type to another in systems programming.

tcg crest

Inventing Harmonious Future

Dr. Laltu Sardar

laltu.sardar@tcgcrest.org

https://laltu-sardar.github.io.