Structured Programming Language

Gourab Saha
Lecturer
CSE, UIU

Ajwad Akil
Lecturer
CSE, UIU

Identifiers and Keywords

1. What is an identifier?
    a. Used to identify variable, function etc.
    b. Case sensitive
2. Keyword
    a. Cannot be used as identifier
    b. e.g int, float, printf, return etc.

Variables

1. Declaration
    a. Type Name;
    b. Eg: **int a;**
    c. Multiple variables of same type: **int a,b,c;**
2. Initialization
    a. **int a = 10;**
3. All Variables Must be defined with a **name** and a **type** before they are use
4. Variables are case sensitive: **a1** and **A1** are **not** the same
5. Long Names: **totalMarks** or **total_marks**
6. Memory for variables

Data Types

Type	Size (bytes)	Format Specifier	Range
int	4 sometimes 2	%d	-2^{15} to $2^{15}-1$ or -2^{31} to $2^{31}-1$
char	1	%c	-128 to 127 or 0 to 255
float	4	%f	3.4-38 to 3.4E+38
double	8	%lf	1.7E-308 to 1.7E+308
short int	2	%hd	
unsigned int	4 sometimes 2	%u	

Data Types(continued)

Type	Size (bytes)	Format Specifier	Range
long int	4 sometimes 2	%ld, %li	
long long int	At least 8	%lld, %lli	
unsigned long int	At least 4	%lu	
unsigned long long int	At least 8	%llu	
signed char	1	%c	
unsigned char	1	%c	

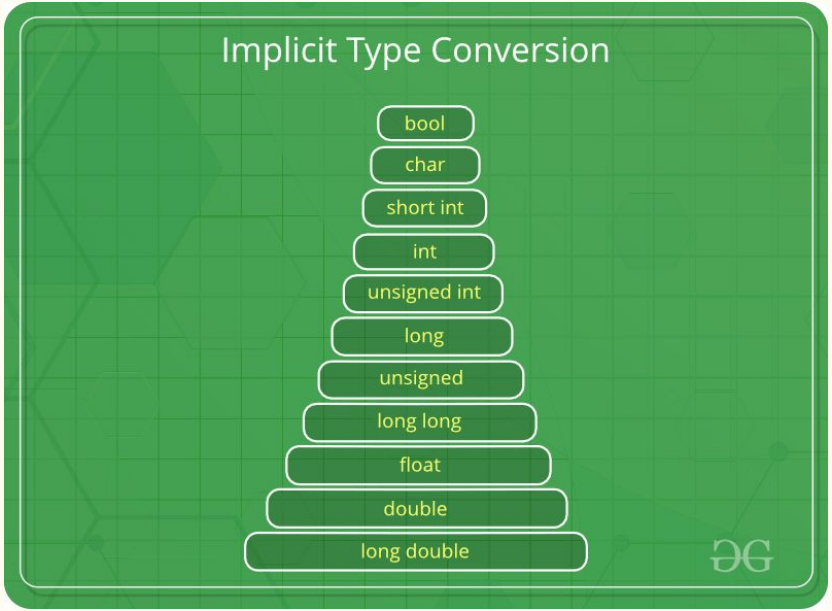
Data Types(continued)

For further reading

1. C - Data Types
2. C Data Types
3. Size of Data Types in C | GATE Notes

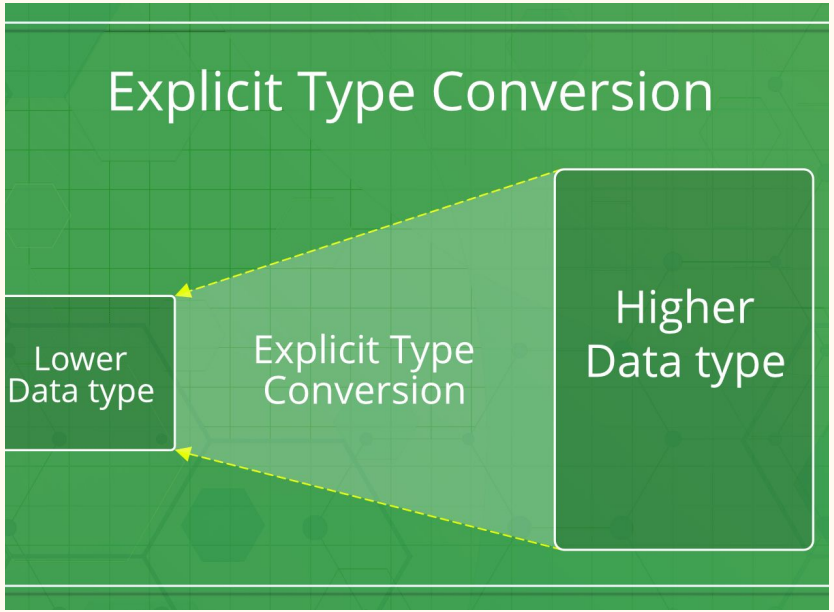
Type Conversion

1. Implicit Type Conversion
   Convert from smaller type to larger type



Type Conversion

1. Explicit Type Conversion - Done by Us
   Convert from Larger type to Smaller type



Type Conversion

1. Implicit Type Conversion
   int a = 10;
   char c = ‘a’;
   int result = a + c;
2. Explicit Type Conversion
   type variable = (**type**) value
   int a = (int) 3.1416

Input and Output - Format Specifiers

1. scanf
   e.g. scanf(“%d”, &age); scanf(“%f”, &length); scanf(“%c”, &a);
2. printf
   e.g. printf(“%d\n”, age); scanf(“%f\n”, length); scanf(“%c\n”, a);

ASCII Values

1. What is the output for the following?
   char c = “a”;
   printf(“%c”, c);
   Ans: a
2. What is the output for the following?
   char c = “a”;
   printf(“%d”, c);
   Ans: 97

ASCII Values

1. Most Common Character Encoding Format for **Text Data** for Computer and the internet
2. 128 alphabetic, numeric, special additional characters and control codes
   eg:
   A = 65
   a = 97
   0 = 48
   9 = 57

For Further Reading:
https://www.rapidtables.com/code/text/ascii-table.html?viewsel=on

Operations in C

Classification based on type of operation:

1. Arithmetic
2. Relational
3. Assignment
4. Bitwise
5. Logical

Classification of number of operands:

1. Binary
2. Unary

Please check the link below:
C - Operators

Arithmetic in C

1. Add +
2. Subtract -
3. Multiply *
4. Divide /
5. Mod %
6. Increment ++
7. Decrement - -

1- 5 are binary and 6-7 are unary

Increment and Decrement Operator

```c
int main(void) {

  int a=0,b=0;

  a++;
  ++b;
  printf("a = %d, b = %d\n\n", a, b);

  int res = a++;
  printf("res = %d\n", res);
  printf("a = %d\n\n", a);

  res = ++b;
  printf("res = %d\n", res);
  printf("b = %d\n", b);
  return 0;
}
```

```
a = 1, b = 1

res = 1
a = 2

res = 2
b = 2
```

Relational

1. Used for comparison
   e.g. ==, <, <=, >, >=, != etc.

Assignment

1. Used for assignment with or without additional operation
   =, +=, -=, *= etc.

Logical

1. Used to perform logical AND, OR, NOT etc.
   e.g. &&, ||, ! etc.

Bitwise

1. Used to perform bitwise operation i.e manipulate data in bit level
   e.g. &, |, ^, >>, << etc.

Bitwise Program

```c
#include <stdio.h>
int main()
{
  int a = 12, b = 25;
  printf("Output a|b = %d\n", a|b);
  printf("Output a&b = %d\n", a&b);
  printf("Output a^b = %d\n", a^b);
  printf("Output ~a = %d\n", ~a);  // ~N = -(N+1

  int num=212;
  printf("Right shift by %d: %d\n", 2, num>>1);
  printf("Right shift by %d: %d\n", 4, num>>2);

  printf("Left shift by %d: %d\n", 2, num<<1);
  printf("Left shift by %d: %d\n", 4, num<<2);
}
```

```
Output a|b = 29
Output a&b = 8
Output a^b = 21
Output ~a = -13
Right shift by 2: 106
Right shift by 4: 53
Left shift by 2: 424
Left shift by 4: 848
```

Other Operators

- * operator, associated with pointers
- ? - Ternary Operator, helps us to branch
  - Expression 1 ? Expression 2 : Expression 3
  - x>y ? printf(“Yes”) : printf(“No”);

Precedence and Associativity of Operators

1. Expressions are evaluated from left to right
2. Operators have precedence
3. Associativity determines order of evaluation of an expression

Please read the following link:
C Operator Precedence - cppreference.com
C Operator Precedence Table
Precedence and order of evaluation | Microsoft Docs
Operator Precedence and Associativity in C - GeeksforGeeks

math.h header function

- ceil(2.5) = 3
- floor(2.5) = 2
- abs(-2) = 2
- sqrt(16) = 4
- pow(2,4) = 16
- sin(), cos(), tan() - takes radian as input, returns double

Branching Statements

- if statements
- if - else if - else statement
- Lets us branch off to different decisions

Problem Solving Session

1. Quadratic Formula Implementation with math header files
2. Odd-Even Integers
3. Vowel or Consonant

- Check LMS for assignment