Basics of Algorithms

Basic Tools for Building Algorithms

- Atomic Data
  - Numbers: integers, real (floating point)
  - Characters: alphabetic and symbols
  - Booleans: true/false
  - Pointers: reference memory addresses (locations)

- Operators
  - Assignment: number1 = number2
  - Arithmetic: +, -, *, /, ^
  - Input/Output: scanf, printf (in C)
  - Relational: <, >, ==, etc.
  - Boolean: and, or, not

Creating Simple Variables

A ***variable*** can be thought of as a named box, or cell, in which one or more data values are stored and may be read or written by the algorithm.

An ***atomic variable*** is a variable that can hold only 1 individual piece of data, such as a number or a character.

The act of creating a variable is called ***declaring the variable***. For every variable that is declared it must be explicitly typed. In other words, each variable has an associated ***data type***.

An ***identifier*** is simply the algorithmic terminology for a name that we "make-up" to "identify" the variable. Every variable must be given a ***unique identifier*** so that there will be no ambiguity as to which piece of data we are referencing.

Examples

```
int age;  /* declares a variable named age whose type is integer */
int test_score;
float  average;  /*declares a variable named average of type float */
```

Rules for Variable Identifiers in C

- A sequence of letters, digits, and the special character _ (underscore).
- A letter or the underscore character must be the first character of an identifier.
- The C language is case-sensitive. This means that *first* and *First* are two different identifiers.

Atomic Data Types in C

- Numbers
  - **int**: 4 bytes
  - **short**: 2 bytes
  - **long**: 4 bytes
  - **unsigned**: 4 bytes
  - **float**: 4 bytes
  - **double**: 8 bytes
  - **long double**: 8 bytes

The range of values that can be stored in each type depends upon the particular computer hardware on which the program will be executed.

- Characters
  - **char**: 1 byte

Examples

```
char  your_grade;
char  first, middle, last;
int age, year;
```

A variable can be assigned an initial value at the time it is declared.

```
int  height = 75;
char grade = 'A';
```

Pointers

Pointers are used to access memory and manipulate memory addresses.

```
int  alpha;    /* alpha is an integer variable */
int  *ptr;     /* ptr is a pointer to an integer  */

alpha = 7;          /* value of 7 is assigned to alpha */
ptr = &alpha;       /* ptr is assigned to reference the location named alpha */
```


```
printf("%d\n", *ptr);  /* prints the contents of the address referenced by ptr */
```

We'll cover pointers in much more detail later, for now just be aware that they are an atomic data type in C.

Arithmetic Operators in C

The four basic arithmetic operators are provided in C along with the modulus operator.

- Addition: +
- Subtraction:
- Multiplication: *
- Division: /
- Modulus: %

Modulus (or modulo) operator produces the remainder portion of division.

For example, 11 % 5 = 1, 20 % 3 = 2 , 3 % 3 = 0

Operator Precedence

Let x = 1 + 2 * 3; What is the value of this expression? I hope you didn't say 9! The answer is clearly 7 since the multiplication operation has precedence over the addition operation.