**KEY CONCEPTS**

**Problem**
Definition of task to be performed (by a computer)

**Algorithm**
A particular sequence of steps that will solve a problem
Steps must be precise and mechanical
The notion of algorithm is a (the?) fundamental intellectual concept associated with computing

**Program**
An algorithm expressed in a specific programming language

**PROBLEM SOLVING**

1. Clearly specify the problem
2. Analyze the problem
3. Design an algorithm to solve the problem
4. Implement the algorithm (write the program)
5. Test and verify the completed program

**WHAT IS A PROGRAMMING LANGUAGE?**

A formal language designed to express computations

Write statements that are translated into machine instructions

**WHAT IS A PROGRAM?**

A sequence of instructions specifying how to perform some task

Break the task into smaller ("unit") subtasks:
- Input: get data from keyboard, file, or other device
- Output: write data to file, screen, or other device
- Math
- Testing: check for a particular condition
- Repetition: perform a particular action repeatedly
Variables

Memory is a collection of locations. The computer attaches the name of a variable to a location in memory.

- **age**: 18
- **good**: true
- **school**: "Tufts"

Types

The type of a variable determines the kind of values it can store and the operations that can be performed on it.

- **int**: age
  - 19
- **boolean**: good
  - true
- **string**: school
  - "Tufts"

Defining Variables

Declare a variable by writing the type followed by the name:

```cpp
int age;
string school;
```

Variables must be declared before they are used.

Naming Variables

In C++, variable names must:

- begin with a letter
- be a sequence of letters, digits, or underscores
- not contain symbols
- not be a reserved word (keyword)
- be unique

The values stored at these locations can be altered...

- **age**: 19
- **good**: true
- **school**: "Tufts"
WHICH OF THESE ARE VALID NAMES?

- totalSales
- total_Sales
- total.Sales
- 4QtrSales
- totalSale$

USING VARIABLES

Remember: A new variable must be **declared** (with its name and type) before being used.

```java
string firstName;
int age;
float GPA;
```

A value can be stored in **assigned** to a declared variable.

```java
firstName = "Jumbo");
age = 18;
GPA = 3.4;
```

USING VARIABLES

A new variable must be **declared** (with its name and type) before being used.

A value can be stored in **assigned** to a declared variable.

Declaration and assignment can happen on the same line.

```java
string firstName;
int age;
float GPA;
firstName = "Jumbo");
age = 18;
GPA = 3.4;
double oldGPA = 3.65;
```

THE **int** DATA TYPE

Designed to hold whole numbers (integers)

Can be signed or unsigned:

```java
int age = 19;
int temperature = -15;
```
ARITHMETIC OPERATORS

Used for performing numeric calculations

C++ has unary and binary operators

Unary: 1 operand
Example: -5

Binary: 2 operands
Example: 13-7

BINARY ARITHMETIC OPERATORS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>Example</th>
<th>ans</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
<td>ans = 7 + 3;</td>
<td>10</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
<td>ans = 7 - 3;</td>
<td>4</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
<td>ans = 7 * 3;</td>
<td>21</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
<td>ans = 7 / 3;</td>
<td>2</td>
</tr>
<tr>
<td>%</td>
<td>modulus</td>
<td>ans = 7 % 3;</td>
<td>1</td>
</tr>
</tbody>
</table>

Remember, this is assuming: int ans;

MATHEMATICAL EXPRESSIONS

An expression can be a constant, a variable, or a mathematical combination of constants and variables.

Create complex expressions using multiple operators

Use expressions in assignments, with other operators, etc.

Examples:

\[ \text{circ} = 2 \times \text{PI} \times \text{radius}; \]
\[ \text{area} = \text{PI} \times \text{radius} \times \text{radius}; \]

OPERATOR PRECEDENCE

Order of operations in a complex expression:
- (unary negation) (in order, left to right)
* / % (in order, left to right)
+ - (in order, left to right)

Example: \(2 + 2 \times 2 - 2 = ?\)

Answer: 4

OPERATOR PRECEDENCE

Order of operations in a complex expression:
- (unary negation) (in order, left to right)
* / % (in order, left to right)
+ - (in order, left to right)

Example: \(2 + 2 \times 2 - 2 = ?\)

Answer: 4

PARENTHESES

Parentheses ( ) override the default precedence order

\(2 + 2 \times 2 - 2 = 4\)
\((2 + 2) \times 2 - 2 = 6\)
\(2 + 2 \times (2 - 2) = 2\)
\((2 + 2) \times (2 - 2) = 0\)

Use parentheses to explicitly set the order of operations
QUESTIONS?

THE char DATA TYPE

Notation: single-quotes

'\n'

The string is comprised of characters between the quotes

CHARACTER STRINGS

Store a series of chars in consecutive memory locations
Notation: double-quotes

"Hello"

The string is comprised of characters between the quotes

QUESTIONS?

CHARACTER STRINGS

Define string variables:

```cpp
string firstName;
string lastName;
```

Assign values:

```cpp
firstName = "George";
lastName = "Washington";
```

THE cout OBJECT

Displays data on the computer screen

Use the binary operator `<<` to send data to cout:

```cpp
cout << "Hello there!";
```
THE `cout` OBJECT

Can send more than one item:

```cpp
cout << "Hello " << "there!";
cout << "Hello... ";
cout << "again,";
```

What is the output on this sequence of instructions?

THE `cout` OBJECT

To print on a new line, use `endl` or the special character `
` in the output string

```cpp
cout << "Hello, there!" << endl;
cout << "Hello, there!\n";
```

What is the output on this sequence?

THE `cin` OBJECT

Reads input from keyboard

Use the binary operator `>>` to retrieve data from `cin`:

```cpp
cin >> age;
```

Where does this data go?

THE `cin` OBJECT

Input data is stored into a variable

Determined by variable types

Example:

```cpp
int age;
cin >> age;
cout << "You typed " << age;
```

QUESTIONS?

PARTS OF A C++ PROGRAM

**Keywords** (reserved)
- Have a special meaning in C++
- Cannot be used for any other purpose

**Programmer-defined symbols**
- Names made up by the programmer, not part of C++
- Used to represent various things: variables, functions, etc.

**Operators**
- Arithmetic
- Assignment

**Punctuation**
THE SIMPLEST PROGRAM

```cpp
// Sample C++ program
#include <iostream>
#include <string>
using namespace std;
int main(){
    cout << "Hello, there!";
    return 0;
}
```

Comment

Preprocessor directives

Namespace to use

Beginning of main function definition

Output statement

Main function returns 0 to the OS

End of main function definition

“BOILERPLATES”

Some parts of the program are standard utterances that need to be included at the beginning and end

```cpp
// Sample C++ program
#include <iostream>
#include <string>
using namespace std;
int main(){
    cout << "Hello, there!";
    return 0;
}
```

Main

Most statements for calculation will appear here (between the curly braces)

Execution begins here

STATEMENTS

Specify actions that will be performed when the program runs

```cpp
// Sample C++ program
#include <iostream>
#include <string>
using namespace std;
int main(){
    cout << "Hello, there!";
    return 0;
}
```

A single statement

A statement must end with a semicolon

QUESTIONS?
A SAMPLE PROBLEM

Problem: Find the size of a rectangular grid

What are the quantities to store?

ANALYSIS

Data types for number of lines and number of columns?

Where do these values come from?
User, file, database, etc.

How is the answer to be reported?
Write to screen, to file, store in database, draw a graph, etc.

DETAILED PROBLEM SPECIFICATION

User provides the dimensions of the grid
(number of rows and columns)

Program provides the size (number of cells)

EXAMPLE TRACE OF INTERACTION

How many lines in the grid? 10
How many columns? 3
The grid size is 30 cells.

(Note: the characters in blue are entered by the user.
You have no control over the formatting of user input.)

ALGORITHM

1. Read in the number of lines
2. Read in the number of columns
3. Multiply number of lines and number of columns
4. Report the answer

IMPLEMENTATION

Write a C++ program that carries out the algorithm

Start with a program sketch
PROGRAM SKETCH

1. Ask the user to enter the number of lines
2. Read the number (call the variable numlines)
3. Ask the user to enter the number of columns
4. Read the number (call the variable numcols)
5. Multiply the number of lines and the number of columns
6. Output the answer

DEBUGGING

Programming is complicated
The errors we make are referred to as bugs
Finding and eliminating them is the process of debugging

TYPES OF ERRORS

Compile-time errors prevent the compiler from successfully translating the program

Compiler can only translate a program with correct syntax

Common syntax errors:
  forgetting a semi-colon or closing brace
  incorrect capitalization

Run-time errors prevent the computer from successfully interpreting the byte code

Logic errors (semantic errors) are errors in the meaning of the program

SUMMARY

Variable and types
Arithmetic operators
Problem solving
Programming

BREAK

Set up CS lab accounts