Computer Science

Foundational programming concepts

Prof. Michel A. Kinsy, Dr. Christopher Dudley, Sena E. Koehnen, Shreeeya Khadka, Dr. Linda Hansen & Rebecca Rodas
Computer Language

- On/Off and the power of 1s and 0s
Computer Language

- On/Off and the power of 1s and 0s
Computer Language

- On/Off and the power of 1s and 0s

<table>
<thead>
<tr>
<th>Division</th>
<th>Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>41 / 2</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>20 / 2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>10 / 2</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>5 / 2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2 / 2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 / 2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Binary number 101001
In Class Exercise

• Binary representation of 24
  ‣ In pencil write in your notebook

• These exercises will be transferred to the computer in the second half of the class
In Class Exercise

• Binary representation of 24
  ‣ In pencil write in your notebook

• Answer
  
  \[
  \begin{align*}
  24/2 &= 12 \text{ remainder } 0 \\
  12/2 &= 6 \text{ remainder } 0 \\
  6/2 &= 3 \text{ remainder } 0 \\
  3/2 &= 1 \text{ remainder } 1 \\
  1/2 &= 0 \text{ remainder } 1 \\
  \end{align*}
  \]

  24 in binary is then 11000
Programming Languages

• There are many programming languages available: Pascal, C, C++, Java, Ada, Perl and Python

• All of these languages share core concepts

• By focusing on these concepts, you are better able to learn any programming language

• Hence, by learning C++, you are poised to learn other languages, such as Java or Perl
  ‣ We will learn through C++
Data types

• Core Concepts
  ‣ There are five basic data types associated with variables:
    • int - integer: a whole number.
    • float - floating point value: i.e. a number with a fractional part.
    • double - a double-precision floating point value.
    • char - a single character.
    • void - valueless special purpose type which we will examine closely in later sections.
Data types

- Data and Operations on that Data
  - In C++
    - Numbers: 5, 7, 3.14, ...
    - Floating-point number: 1.234
    - Integer: 1234
    - Booleans: true or false
    - Characters: 'a', 'b', ... 'z', '1', '2', ... '9', '!', '^', ..."
Data types

• In C++
  ‣ Arrays (a list of data (all of the Same Data Type!))
    • int grades = [94, 78, 88, 90, 93, 87, 59];
  ‣ Structures (a collection of named data referring to a single entity)

```c
struct Student {
    string name;
    int id;
    int age;
    char cohort [10];
};
```
In Class Exercise

• Declare a variable and assign to it a value
  ‣ In pencil write in your notebook

• These exercises will be transferred to the computer in the second half of the class
In Class Exercise

• Declare a variable and assign to it a value
  ‣ In pencil write in your notebook

• Answer

  int forward = 2;

  int movement;

  movement = forward;

• These exercises will be transferred to the computer in the second half of the class
In Class Exercise

• Declare a variable and assign to it a value
  ‣ In pencil write in your notebook

• Answer

\[
\text{int } \text{forward} = 2
\]

• These exercises will be transferred to the computer in the second half of the class
#include <iostream>
#include <string>
using namespace std;

int main(void) {
    Student s1;
    cout << "What's your name? ";
    getline(cin, s1.name);
    cout << "Enter id: ";
    cin >> s1.id;
    cout << "Enter age: ";
    cin >> s1.age;
    cout << "Enter cohort: ";
    cin.get(p1.name, 10);
    cout << \nDisplaying Student Information." << endl;
    cout << "Name: " << s1.name << endl;
    cout << "ID: " << s1.id << endl;
    cout << "Age: " << s1.age << endl;
    cout << "Cohort: " << s1.cohort;
    return 0;
}
#include <iostream>
#include <string>
using namespace std;

int main(void) {
    Student s1;
    cout << "What's your name? ";
    getline (cin, s1.name);
    cout << "Enter id: ";
    cin >> s1.id;
    cout << "Enter age: ";
    cin >> s1.age;
    cout << "Enter cohort: ";
    cin.get(p1.name, 10);
    cout << "nDisplaying Student Information." << endl;
    cout << "Name: " << s1.name << endl;
    cout << "ID: " << s1.id << endl;
    cout << "Age: " << s1.age << endl;
    cout << "Cohort: " << s1.cohort;
    return 0;
}
Conditional execution

• C++ contains a type named bool
  ‣ true
  ‣ False

• Boolean operators
  ‣ The and operator is &&
  ‣ The or operator is ||
  ‣ The not operator is !

• Equality operators
  ‣ ==  !=
Conditional execution

- Conditional constructs provide the ability to control whether a statement list is executed

- If statement
  - If
    - if (Expression)
      - Action

Expression

true

false

Action
Conditional execution

- Conditional constructs provide the ability to control whether a statement list is executed

- If statement
  - If
    - if (Expression)
    - Action

```cpp
cout << "Enter two integers: ";
Int num1;
Int num2;
cin >> num1 >> num2;
if (num1 > num2) {
    int remember_num1 = num1;
    num1 = num2;
    num2 = remember_num1;
}
cout << "Inputs in sorted order:
<< num1 << " " << num2 << endl;
```
In Class Exercise

• Write a program so that if button 2 is pressed the robot goes forward
  ‣ In pencil write in your notebook

• These exercises will be transferred to the computer in the second half of the class
In Class Exercise

• Write a program so that if button 2 is pressed the robot goes forward
  ‣ In pencil write in your notebook

• Answer

  \[\text{If (button == 2)}\]
  \[\text{movement = forward;}\]

• These exercises will be transferred to the computer in the second half of the class
Conditional execution

- Conditional constructs provide the ability to control whether a statement list is executed

- If statement
  - if-else
    - if (Expression)
      - Action1
    - else
      - Action2

Conditional execution

- Conditional constructs provide the ability to control whether a statement list is executed

- If statement
  - if-else
    - if (Expression)
      - Action1
    - else
      - Action2

```
cout << "Enter two integers: ",
Int num1;
Int num2;
cin >> num1 >> num2;
Int max;
if (num1 < num2) {
    max = num2;
} else {
    max = num1;
}
cout << "Maximum of inputs is: " << max << endl;
```
In Class Exercise

• Write a program so that if button 2 is pressed the robot goes forward else it stops
  ‣ In pencil write in your notebook

• These exercises will be transferred to the computer in the second half of the class
In Class Exercise

• Write a program so that if button 2 is pressed the robot goes forward else it goes left
  ‣ In pencil write in your notebook

• Answer

    If (button == 2)
      movement = forward;
    else
      movement = left;
Conditional execution

• For loop
• Do loop
• While loop
• Do while loop
Iteration

• For loops
  ‣ The general format when using for loops is
    • for ( initialization; LoopContinuationTest; increment )

Statement

Initialize variable

Condition
  Test the variable

true
  statement
  Increment variable

false
Iteration

• For loops

  ‣ The general format when using for loops is

  • for ( initialization; LoopContinuationTest; increment )
    Statement

for( int counter = 1; counter <= 10; counter++ )
cout << counter << endl;
In Class Exercise

• Write a program so that if button 1-4 are pressed the robot makes that many moves then stops
  ‣ In pencil write in your notebook

• These exercises will be transferred to the computer in the second half of the class
In Class Exercise

• Write a program so that if button 1-4 are pressed the robot makes that many moves then stops
  ‣ In pencil write in your notebook

• Answer

  \[
  \text{for(int } i = 0; i < \text{button}; i++)
  \]
  \[
  \text{movement} = \text{move};
  \]
Iteration

• While loop
  ‣ while loop repeated until condition becomes false

  • Initialization;
    while ( loopContinuationTest){
      statement
      increment;
    }

Iteration

- While loop
  - while loop repeated until condition becomes false

- Initialization;

```cpp
while (loopContinuationTest) {
  statement
  increment;
}
```

```cpp
int counter = 1;     //initialization
while (counter <= 10) { //repetition
  condition
  cout << counter << endl;
  counter++;          //increment
}
```
Iteration

- Do while
  - do/while repetition structure is similar to the while structure
  - Condition for repetition tested after the body of the loop is executed

- do {
  statement
} while ( condition );
In Class Exercise

• Write a program so that if the play button is pressed the robot goes in square movements until you pressed stop
  ‣ In pencil write in your notebook
In Class Exercise

• Write a program so that if the play button is pressed the robot goes in square movements until you pressed stop
  ‣ In pencil write in your notebook

• Answer

```
    do
        movement = move;
    while (button != stop)
```

Recursion

• Sometimes, the best way to solve a problem is by solving a smaller version of the exact same problem first

• Recursion is a technique that solves a problem by solving a smaller problem of the same type
Recursion

- Sometimes, the best way to solve a problem is by solving a smaller version of the exact same problem first.
- Recursion is a technique that solves a problem by solving a smaller problem of the same type.
- The technique ends up with functions that call themselves (recursive functions).
Factorial

• \( n! \) is read \( n \) factorial

• \( n! = 1 \times 2 \times 3 \times \ldots \times n \)
  
  › \( 0! = 1 \)
  
  › \( 1! = 1 \)

  › \( 2! = 1 \times 2 = (1!) \times 2 \)

  › \( 3! = 1 \times 2 \times 3 = (2!) \times 3 = 6 \)

  › \( 4! = 1 \times 2 \times 3 \times 4 = (3!) \times 4 = 24 \)
Recursion

• Factorial function
  ‣ Iterative implementation

```c
int Factorial(int n)
{
    int fact = 1;
    for(int count = 2; count <= n; count++)
        fact = fact * count;
    return fact;
}
```
Recursion

• Factorial function
  ‣ Recursive implementation

```c
int Factorial(int n) {
    if (n==0) // base case
        return 1;
    else
        return n * Factorial(n-1);
}
```
Next Class Objects