CSC 101 Lecture Notes Week 1

Intro to the Course
Intro to Programming and Problem Solving

I. Introductory course materials

I. Introductory course materials

A. Course syllabus

I. Introductory course materials

A. Course syllabus

B. Lecture notes week 1 (these notes)

I. Introductory course materials

A. Course syllabus

B. Lecture notes week 1 (these notes)

C. Lab 1 writeup

I. Introductory course materials

A. Course syllabus

B. Lecture notes week 1 (these notes)

C. Lab 1 writeup

D. Program 1 writeup

II. What is a program?

II. What is a program?

A. In simple terms, it’s a list of instructions.

II. What is a program?

A. In simple terms, it’s a list of instructions.

B. In this sense, humans follow programs frequently,

II. What is a program?

A. In simple terms, it’s a list of instructions.

B. In this sense, humans follow programs frequently,

1. a cooking recipe

II. **What is a program?**

A. In simple terms, it’s a list of instructions.

B. In this sense, humans follow programs frequently,

1. a cooking recipe

2. a set of directions to get to someone’s house

What is a program, cont’d

C. Programs for computers differ from humans’

What is a program, cont’d

C. Programs for computers differ from humans’

1. Written in vastly simpler language

What is a program, cont'd

C. Programs for computers differ from humans'

1. Written in vastly simpler language

a. Humans communicate to one another using ***natural languages***

What is a program, cont’d

C. Programs for computers differ from humans’

1. Written in vastly simpler language

   a. Humans communicate to one another using ***natural languages***

   b. Humans must communicate programs using ***programming languages***

What is a program, cont’d

2. Computer programs must be 100% ***grammatically correct***

What is a program, cont’d

2. Computer programs must be 100% ***grammatically correct***

    a. Humans communicate ungrammatically

What is a program, cont'd

2. Computer programs must be 100% ***grammatically correct***

   a. Humans communicate ungrammatically

   b. Can't happen in computer code

What is a program, cont’d

2. Computer programs must be 100% ***grammatically correct***

   a. Humans communicate ungrammatically

   b. Can’t happen in computer code

   c. Mundane and annoying part of programming

III. What is problem solving?

A. Three major phases:

III. **What is problem solving?**

A. Three major phases:

1. Stating ***what the problem is***

III. What is problem solving?

A. Three major phases:

1. Stating ***what the problem is***

2. Defining ***how to solve the problem***

III. What is problem solving?

A. Three major phases:

1. Stating ***what the problem is***

2. Defining ***how to solve the problem***

3. Verifying ***that the solution is correct***

What is problem solving, cont’d

B. Humans solve problems with vast knowledge

What is problem solving, cont’d

B. Humans solve problems with vast knowledge

C. Computers have much less knowledge

What is problem solving, cont'd

B. Humans solve problems with vast knowledge

C. Computers have much less knowledge

D. In a human/computer problem solving team, the computer is the ***junior partner***

Junior partner, cont’d

1. Human states the problem

Junior partner, cont'd

1. Human states the problem

2. Human defines the solution

Junior partner, cont’d

1. Human states the problem
2. Human defines the solution
3. Human writes the program

Junior partner, cont’d

1. Human states the problem
2. Human defines the solution
3. Human writes the program
4. Computer compiles the program

Junior partner, cont’d

1. Human states the problem
2. Human defines the solution
3. Human writes the program
4. Computer compiles the program
5. Computer runs the compiled program

Junior partner, cont’d

1. Human states the problem
2. Human defines the solution
3. Human writes the program
4. Computer compiles the program
5. Computer runs the compiled program
6. Human validates that the answer is correct

IV. What is a computer?

IV. What is a computer?

A. An electronic device that can follow programmed instructions in ***machine language***

IV. What is a computer?

A. An electronic device that can follow programmed instructions in ***machine language***

1. Machine language is simpler than C

IV. **What is a computer?**

A. An electronic device that can follow programmed instructions in ***machine language***

1. Machine language is simpler than C

2. It’s stored in binary computer memory

What is a computer, cont’d

B. Major components of a computer:

What is a computer, cont’d

B. Major components of a computer:

1. Central processing unit (CPU)

What is a computer, cont'd

B. Major components of a computer:

1. Central processing unit (CPU)

2. Memory unit

What is a computer, cont’d

B. Major components of a computer:

1. Central processing unit (CPU)
2. Memory unit
3. Peripheral memory

What is a computer, cont'd

B. Major components of a computer:

1. Central processing unit (CPU)
2. Memory unit
3. Peripheral memory
4. Peripheral input and output devices

What is a computer, cont’d

C. For CSC 101, we will delve no further

What is a computer, cont’d

C. For CSC 101, we will delve no further

1. A ***compiler*** translates C into machine language.

What is a computer, cont’d

C. For CSC 101, we will delve no further

1. A ***compiler*** translates C into machine language.

2. Compiler and computer are "black boxes".

What is a computer, cont’d

C. For CSC 101, we will delve no further

1. A ***compiler*** translates C into machine language.
2. Compiler and computer are "black boxes".
3. Your job in 101 is to solve problems in C.

What is a computer, cont’d

4. You as the programmer ***take on faith***
   a. that program is compiled correctly
   b. that the computer works correctly

V. On natural & programming languages

A. Why do we use the languages that we do?

V. On natural & programming languages

A. Why do we use the languages that we do?

1. Why is English a dominant natural language?

V. On natural & programming languages

A. Why do we use the languages that we do?

1. Why is English a dominant natural language?

2. Why is C a dominant programming language?

Languages cont’d

B. Popularity may have little to do with quality.

Languages cont’d

B. Popularity may have little to do with quality.

1. Is English the "best" natural language?

Languages cont'd

B. Popularity may have little to do with quality.

1. Is English the "best" natural language?

2. Is C the "best" programming language?

Languages cont'd

B. Popularity may have little to do with quality.

1. Is English the "best" natural language?

2. Is C the "best" programming language?

3. Answer to both is ***probably not!***

Languages cont'd

B. Popularity may have little to do with quality.

1. Is English the "best" natural language?
2. Is C the "best" programming language?
3. Answer to both is ***probably not!***
4. However, we gotta live with them both :(

VI. A simple introductory problem -- is a number positive or negative?

VI. A simple introductory problem -- is a number positive or negative?

A. Consider how to solve as a human.

VI. A simple introductory problem -- is a number positive or negative?

A. Consider how to solve as a human.

1. Seems like a pretty darn simple problem.

VI. A simple introductory problem -- is a number positive or negative?

A. Consider how to solve as a human.

1. Seems like a pretty darn simple problem.

2. The solution goes something like this:

"Look at a number and tell me if it’s positive".

Simple introductory problem, cont’d

B. To solve with a program, we need to use simpler language, and address questions like this:

Simple introductory problem, cont'd

B. To solve with a program, we need to use simpler language, and address questions like this:

1. *Are we clear what "positive" means?*

Simple introductory problem, cont’d

B. To solve with a program, we need to use simpler language, and address questions like this:

1. *Are we clear what "positive" means*

2. *What do you mean by "look at"?*

Simple introductory problem, cont’d

B. To solve with a program, we need to use simpler language, and address questions like this:

1. *Are we clear what "positive" means?*

2. *What do you mean by "look at"?*

3. *How do you want me to "tell you" the answer?*

Simple introductory problem, cont’d

C. That is, we must ***specify*** the problem clearly.

Simple introductory problem, cont’d

C. That is, we must ***specify*** the problem clearly.

D. Then we must write an ***algorithm*** to solve it.

Simple introductory problem, cont’d

C. That is, we must ***specify*** the problem clearly.

D. Then we must write an ***algorithm*** to solve it.

E. Here’s an example algorithm, in a language a bit simpler than C:

Simple introductory problem, cont’d

```
    begin
```

Simple introductory problem, cont’d

```
begin
  let x be an integer variable
```

Simple introductory problem, cont’d

```
begin
    let x be an integer variable
    read x
```

Simple introductory problem, cont’d

```
begin

  let x be an integer variable

  read x

  if x > 0 then
    print "yes"
```

Simple introductory problem, cont’d

```
begin

  let x be an integer variable

  read x

  if x > 0 then
    print "yes"
  else
    print "no"
```

Simple introductory problem, cont’d

```
begin

  let x be an integer variable

  read x

  if x > 0 then
    print "yes"
  else
    print "no"

end
```

Simple introductory problem, cont’d

F. Here’s the program in C:

C Code:	Algorithm:
`#include <stdio.h>`	*before beginning*

C Code:	**Algorithm:**
`#include <stdio.h>`	*before beginning*
`int main() {`	*begin*

C Code:

```
#include <stdio.h>

int main() {

    int x;
```

Algorithm:

before beginning

begin

let x be an int var

C Code:

```c
#include <stdio.h>

int main() {

    int x;

    scanf("%d", &x);
```

Algorithm:

```
before beginning

begin

let x be an int var

read x
```

C Code:

```c
#include <stdio.h>
int main() {
    int x;
    scanf("%d", &x);
    if (x > 0)
```

Algorithm:

```
before beginning
begin
let x be an int var
read x
if x > 0 then
```

C Code:

```
#include <stdio.h>

int main() {

    int x;

    scanf("%d", &x);

    if (x > 0)
        printf("yes");
```

Algorithm:

```
before beginning

begin

let x be an int var

read x

if x > 0 then
    print "yes"
```

C Code:

```
#include <stdio.h>

int main() {

    int x;

    scanf("%d", &x);

    if (x > 0)
        printf("yes");
    else
```

Algorithm:

```
before beginning

begin

let x be an int var

read x

if x > 0 then
    print "yes"
else
```

C Code:

```
#include <stdio.h>

int main() {

    int x;

    scanf("%d", &x);

    if (x > 0)
        printf("yes");
    else
        printf("no");
```

Algorithm:

```
before beginning

begin

let x be an int var

read x

if x > 0 then
    print "yes"
else
    print "no"
```

C Code:

```
#include <stdio.h>

int main() {

    int x;

    scanf("%d", &x);

    if (x > 0)
        printf("yes");
    else
        printf("no");

}
```

Algorithm:

```
before beginning

begin

let x be an int var

read x

if x > 0 then
    print "yes"
else
    print "no"

end
```

Simple introductory problem, cont’d

G. Here are some fundamental aspects of a program:

Simple introductory problem, cont’d

G. Here are some fundamental aspects of a program:

1. Programs have an explicit ***beginning and ending***

Simple introductory problem, cont'd

G. Here are some fundamental aspects of a program:

1. Programs have an explicit ***beginning and ending***

2. Programs use ***variables***

Simple introductory problem, cont’d

G. Here are some fundamental aspects of a program:

1. Programs have an explicit ***beginning and ending***
2. Programs use ***variables***
3. Programs precisely define ***input and output***

Simple introductory problem, cont’d

G. Here are some fundamental aspects of a program:

1. Programs have an explicit ***beginning and ending***
2. Programs use ***variables***
3. Programs precisely define ***input and output***
4. Programs have ***arithmetic expressions***

Simple introductory problem, cont’d

5. A fundamental construct is the ***conditional***.

Simple introductory problem, cont’d

5. A fundamental construct is the ***conditional***.

   a. Common syntax is an "if" statement.

Simple introductory problem, cont’d

5. A fundamental construct is the ***conditional***.

   a. Common syntax is an "if" statement.

   b. Used above to decide whether x > 0.