CIS 343 - Structure of Programming Languages

Class Overview

Ira Woodring

(Follows the Sebesta Text, Chapter 1)

Overview

What's the point?

- Why do we study programming languages?

Overview

Languages give us the ability to express what we are thinking.

Overview

- It can be hard to convey something we don't have a direct translation for.
- Using English for instance, how do you describe:

The wordless yet meaningful look shared by two people who desire to initiate something, but are both reluctant to start.

Mamihlapinatapei

A Yagan word (a native tribe from Tierra Del Fuego).

A relationship by fate or destiny.

Yuanfen

Chinese

The act of tenderly running your fingers through someone's hair.

Cafuné

Brazilian Portuguese

The happiness of meeting again after a long time.

Retrouvailles

French

A person who is willing to forgive abuse the first time; tolerate it the second time, but never a third time.

Ilunga

Bantu

The heart-wrenching pain of wanting someone you can’t have.

La Douleur Exquise

French

The sense upon first meeting a person that the two of you are going to fall into love.

Koi No Yokan

Japanese

A declaration of one’s hope that they’ll die before another person, because of how difficult it would be to live without them.

Ya’aburnee

Arabic

The euphoria you experience when you’re first falling in love.

Forelsket

Norwegian

The feeling of longing for someone that you love and is lost. Another linguist describes it as a "vague and constant desire for something that does not and probably cannot exist."

Saudade

Portuguese

Overview

All of these were from http://bigthink.com/harpys-review/the-top-10-relationship-words-that-arent-translatable-into-english

Overview

On a similar note, the Sami people of Scandinavia and Russia have over **180 words for snow**.

- They have so many different ways to express snow, each with subtle nuances so that they can perfectly express ideas!
- Let's consider a computer example:

Overview

What does this code do?

```
mov eax, $x                  // Move the value in $x to the register eax
beginning:
cmp eax, 0x0A                // Compare the value in eax to 0x0A (10)
jg end                       // If they are the same, go to end.
inc eax                      // Increase the value in eax
jmp beginning                // Go to the beginning
end:
mov $x, eax                  // Move the value in eax to $x
```

Overview

It is the same as this!

```
while(x <= 10){
  x++;
}
```

Why was it so much more complicated in the first example??

Overview

Assembly language lacks the concept of looping!

- Looping is a higher-order idea.
- Higher-order languages allow us to express these ideas more easily and succinctly.

Overview

The tools we are provided with help us "attack the problem".

Overview

- i.e., you *can* use a wrench as a hammer, but doesn't mean it is ideal (or that you should...)
- For instance, some people may want to write web programs with HTML forms via C.

(taken from https://www.cs.tut.fi/~jkorpela/forms/cgic.html)

-- Show code sample --

Overview

Code is a nightmare!

- Relies on hard coding some values sent, such as headers. This can cause problems as things change.
- C doesn't natively understand the web. You will need to re-invent the wheel continuously to get anything done - even then your code is not likely to be safe.

Compare:

https://www.w3schools.com/php/php_forms.asp

Overview

We *can* use any language we want, but it doesn't mean we should.

- By using C we can't take advantage of well-written, robust methods that other languages may provide for web programming.
- Development and maintenance time will go up.

Overview

Programming languages address specific domains.

- No language is perfect for every domain.
- Studying languages help us to choose the most appropriate language for a job.

For instance, here are Python and C++ samples for how to read a file and print each line:

Python

```
textFile = open(”filename.txt”, ”r”)
lines = textFile.readlines()
for l in lines:
    print l
```

C++

```
#include <fstream>
#include <string>

int main(int argc, char** argv)
{
   std::ifstream file(“filename.txt”);
   std::string str;
   while(std::getline(file, str)){
      std::cout << str << std::endl;
   }
}
```

Overview

Python seems to be more English-like, while C++ includes some constructs that may be confusing.

- What is "::" for instance?
- C++ has a focus on speed and low-level hardware control.
- Python is a more general-purpose language for the masses.

So what domains are there?

Science

- Must be fast
- Works with a lot of floating point numbers
- Must be precise

Business

- Emphasis is on reporting and output
- Lots of records and data

Artificial Intelligence

- Primary data structure may be lists, facts, or other special types.
- Symbolic

Systems Programming

- Must be fast
- Efficient
- Focuses on hardware

Many, many other specialized domains

- scripting tools
- file parsing
- custom tasks

Overview

Also gives us a better ability to express ideas.

- In C (for instance) we don't have a great way to deal with matrices.
- Ruby (and many, many other languages) provide us with custom classes and overloaded operators that make working with such constructs a breeze.

In C, pretending we have a multidimensional array or grid is convoluted:

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv){
  int height = atoi(argv[1]);
  int width = atoi(argv[2]);
  int* matrix = malloc(width * height * sizeof(int));
  for(int i=0; i<height; i++){
    for(int j=0; j<width; j++){
      matrix[i][j] = 0;
    }
  }
}
```

Overview

Ruby (much like many higher order languages) provides easeir to use facilities:

```ruby
require "Matrix"

h=Integer(ARGV[0])
w=Integer(ARGV[1])

m = Matrix.zero(h, w)
```

Overview

Studying languages makes it easier for us to learn new languages.

Overview

Most of you should know Java; with only your Java knowledge you can probably still guess what this Python code is doing:

```python
class House:
  def __init__(self, s, c, p):
    self.sq_ft = s
    self.color = c
    self.price = p
```

Overview

It is the same as this:

```java
public class House {
  public int sq_ft;
  public Color color;
  public float price;

  public House(int s, Color c, float p){
    this.sq_ft = s;
    this.color = c;
    this.price = p;
  }
}
```

Overview

It helps us understand why languages are implemented the way they are.

Consider include guards:

```
...
#ifndef _UNISTDIO_H
#define _UNISTDIO_H

#include "unitypes.h"

/* Get size_t.  */
#include <stddef.h>

/* Get FILE.  */
#include <stdio.h>

/* Get va_list.  */
#include <stdarg.h>

#ifdef __cplusplus
extern "C" {
#endif

/* These work like the printf function family.
...
```

They are in almost every *.h file.

- Why are they there?
- They prevent multiple sourcings of the same code, as the compiler attempts to include each file each time it is needed.
- This is because C compiles **every file separately**.

Overview

It helps us to use languages better.

Overview

In C for instance, it is very common to see folks just exit or return 0 when an error occurs.

- But, if we understand return codes and what they can be used for (automating code runs), we know we can do something better:

```c
#include <stdio.h>
#include <errno.h>

int main(int argc, char** argv){
  FILE *file_descriptor;
  file = fopen("myFile.txt", "rw");
  if(!file){
    return errno;
  }
}
```

Not complete; just an example!

Overview

Helps us advance the field.

Overview

- By understanding what has been created before, we have a better idea of what we don't need to recreate, what we can make better, and what we still need to create.