Advanced Programming Principles CM20214/CM20221

Russell Bradford

2016

Material Outline

This is a Year-Long Unit, meaning that it continues throughout both Semester 1 and Semester 2

Material Outline

This is a Year-Long Unit, meaning that it continues throughout both Semester 1 and Semester 2

The good news is that there is just one exam for the Unit at the end of Semester 2

Material Outline

This is a Year-Long Unit, meaning that it continues throughout both Semester 1 and Semester 2

The good news is that there is just one exam for the Unit at the end of Semester 2

The bad news is that means you won’t be able to get away with cramming the night before the exam

Material Outline

There are two Unit codes: CM20214 and CM20221

Material Outline

There are two Unit codes: CM20214 and CM20221

CM20214 is for “Computer Scientists”, while CM20221 is for “Mathematicians”

Material Outline

There are two Unit codes: CM20214 and CM20221

CM20214 is for “Computer Scientists”, while CM20221 is for “Mathematicians”

The content and exam is identical for both; there is a slight variation in the coursework to accommodate the Integrative Project that CS takes

CM20214/221

The first coursework will be set on this Semester's part of the Unit (10% unit total for CM20214; 20% unit total for CM20221) in particular programming in the **functional style** using **Lisp**

CM20214/221

The first coursework will be set on this Semester's part of the Unit (10% unit total for CM20214; 20% unit total for CM20221) in particular programming in the **functional style** using **Lisp**

Another coursework, worth 20% for everybody, will be set next Semester

CM20214/221

The functional style is notorious as something that many people find very hard to grasp until it “clicks”

CM20214/221

The functional style is notorious as something that many people find very hard to grasp until it “clicks”

When it clicks it becomes a very powerful and widely applicable way of thinking

CM20214/221

The functional style is notorious as something that many people find very hard to grasp until it “clicks”

When it clicks it becomes a very powerful and widely applicable way of thinking

Lisp is chosen as the language to introduce the functional style of programming for many reasons, historical importance being just one

CM20214/221

The functional style is notorious as something that many people find very hard to grasp until it “clicks”

When it clicks it becomes a very powerful and widely applicable way of thinking

Lisp is chosen as the language to introduce the functional style of programming for many reasons, historical importance being just one

Because the functional style is (initially) hard, many people, by transference, think Lisp is very hard

CM20214/221

In fact, Lisp is very simple, it’s just that some people take a long time to realise this

CM20214/221

In fact, Lisp is very simple, it's just that some people take a long time to realise this

But, just as with any other new programming language, the best way to learn is to practice by writing code

CM20214/221

In fact, Lisp is very simple, it's just that some people take a long time to realise this

But, just as with any other new programming language, the best way to learn is to practice by writing code

Early warning: you *must* work for yourself though a large number of Lisp exercises as practice first or you will fail miserably when you try the coursework

CM20214/221

In fact, Lisp is very simple, it's just that some people take a long time to realise this

But, just as with any other new programming language, the best way to learn is to practice by writing code

Early warning: you *must* work for yourself though a large number of Lisp exercises as practice first or you will fail miserably when you try the coursework

Practice practice practice is the best way of getting there

Standard Introductory Slide

Remember:

Standard Introductory Slide

Remember:

You are expected to do some work outside of lectures

Standard Introductory Slide

Remember:

You are expected to do some work outside of lectures

Lectures are the *start* of the learning process, not the end!

Standard Introductory Slide

Remember:

You are expected to do some work outside of lectures

Lectures are the *start* of the learning process, not the end!

These slides *do not* contain all the material you need for the unit: they are notes to me to what to say in lectures

Standard Introductory Slide

Remember:

You are expected to do some work outside of lectures

Lectures are the *start* of the learning process, not the end!

These slides *do not* contain all the material you need for the unit: they are notes to me to what to say in lectures

Do not rely on my notes for your revision

Standard Introductory Slide

Remember:

You are expected to do some work outside of lectures

Lectures are the *start* of the learning process, not the end!

These slides *do not* contain all the material you need for the unit: they are notes to me to what to say in lectures

Do not rely on my notes for your revision

People who do this live to regret it

Standard Introductory Slide

You need to take your own notes, read books, and *participate*

Standard Introductory Slide

You need to take your own notes, read books, and *participate*

You don’t expect to get fit simply by paying to joining a gym. . .

“If you have college courses in CS, buy the books and spend day and night the few days before class going through the books and taking notes and answering questions and programming examples before the first class even starts. If you really want to do this in your life, that’s what you should do, not just wait for the education to be handed you. Those who finish at the top will always be in high demand. You can learn outside of school too but you have to put a lot of time into it. It doesn’t come easily. Small steps, each improving on the other, is what to expect, not instant understanding and expertise.”

Steve Wozniak, co-founder of Apple

CM20214/221

Three hours of lectures a week:

- Monday 9.15 and 10.15
- Friday 11.15

CM20214/221

Coursework timeline (subject to change):

1. set Mon 9 Nov 2015
   due Fri 11 Dec 2015

Feedback on coursework will be provided via Moodle. There will be general feedback that applies to many people and some individual feedback

Unit Outline

Week 6 will be a “consolidation week”

Unit Outline

Week 6 will be a “consolidation week”

No lectures across the Department

Unit Outline

Week 6 will be a “consolidation week”

No lectures across the Department

For the whole of Computer Science (CM Units)

Unit Outline

Week 6 will be a “consolidation week”

No lectures across the Department

For the whole of Computer Science (CM Units)

Presumably other Departments will carry on as usual

Material Outline

In this Semester I shall cover several major chunks of material:

Material Outline

In this Semester I shall cover several major chunks of material:

1. Introduction to C

Material Outline

In this Semester I shall cover several major chunks of material:

1. Introduction to C
2. Functional Style of programming using Lisp

Material Outline

In this Semester I shall cover several major chunks of material:

1. Introduction to C
2. Functional Style of programming using Lisp
3. Comparative languages

Material Outline

In this Semester I shall cover several major chunks of material:

1. Introduction to C
2. Functional Style of programming using Lisp
3. Comparative languages
4. Introduction to AI (time permitting)

Material Outline

In this Semester I shall cover several major chunks of material:

1. Introduction to C
2. Functional Style of programming using Lisp
3. Comparative languages
4. Introduction to AI (time permitting)

In the second Semester Alessio Guglielmi will be covering Logic Programming, grammars and compilation

Material Outline

The main idea of this Unit is to show you many kinds of language and many kinds of programming, to equip you with the means to make the choice of the right tool for the job

Material Outline

The main idea of this Unit is to show you many kinds of language and many kinds of programming, to equip you with the means to make the choice of the right tool for the job

So you don’t try to solve every problem with a Java- or Python-shaped mallet

Material Outline

Hidden within the various languages that exist today, are *a set of paradigms that can completely change the way you are used to thinking*. Sometimes these paradigms are so focused and so specific to a language that they are only applicable in that particular language. Other times I find, and this is the great part; that you can **take those paradigms and apply them to the languages you currently utilize**. When that happens, congratulations, you've expanded your mind and your skill set and additionally you now have a fresh way of tackling stale old problems.

Ralph Caraveo III

Books

The material I shall cover is “mature”, meaning it’s been around for a long time

Books

The material I shall cover is “mature”, meaning it’s been around for a long time

So there are lots of books out there and to some extent it’s a matter of finding a book that suits you

Books

The material I shall cover is “mature”, meaning it’s been around for a long time

So there are lots of books out there and to some extent it’s a matter of finding a book that suits you

Web-based material tends to be fairly accurate: Wikipedia is pretty good in this area

Books

The material I shall cover is “mature”, meaning it’s been around for a long time

So there are lots of books out there and to some extent it’s a matter of finding a book that suits you

Web-based material tends to be fairly accurate: Wikipedia is pretty good in this area

But, as always with Wikipedia, you should treat it as a start and *follow up the references*

Books

For C there are too many books, many quite poor on technical details

Books

For C there are too many books, many quite poor on technical details

There are loads that are OK for an introduction: it’s not too important which you use

Books

For C there are too many books, many quite poor on technical details

There are loads that are OK for an introduction: it’s not too important which you use

But make sure the book covers ANSI C, not K&R (Kernighan and Richie) C, which is an earlier, obsolete standard for C

Books

You might like to look at books on Algorithms that use C, e.g.,

- “Understanding Algorithms and Datastructures” Brunskill and Turner. Uses C and Ada
- “Algorithms in C” Sedgewick
- “Data Structures, Algorithms & Software Principles in C” Standish.

CM20214/CM20221

There is a Web page that contains bits and pieces relevant to this unit:

`http://people.bath.ac.uk/masrjb/CourseNotes/cm20214.html`

CM20214/CM20221

There is a Web page that contains bits and pieces relevant to this unit:

`http://people.bath.ac.uk/masrjb/CourseNotes/cm20214.html`

These notes will be available from there *after* I have covered them in lectures: there is no substitute to having the material go through your brain at least once!

CM20214/CM20221

There is a Web page that contains bits and pieces relevant to this unit:

`http://people.bath.ac.uk/masrjb/CourseNotes/cm20214.html`

These notes will be available from there *after* I have covered them in lectures: there is no substitute to having the material go through your brain at least once!

Note: these notes are for my benefit, to remind me what to say. They do not contain everything I shall say in the the lectures

CM20214/CM20221

Start of the material!

So far you have seen a limited number of styles of programming:

CM20214/CM20221

Start of the material!

So far you have seen a limited number of styles of programming:

- Object Oriented style programming: Java and others

CM20214/CM20221

Start of the material!

So far you have seen a limited number of styles of programming:

- Object Oriented style programming: Java and others
- No style: unstructured things like Basic and assembler

CM20214/CM20221

Start of the material!

So far you have seen a limited number of styles of programming:

- Object Oriented style programming: Java and others
- No style: unstructured things like Basic and assembler

Later you will see the Logic/Declarative style

So far you have seen a limited number of styles of programming:

- Object Oriented style programming: Java and others
- No style: unstructured things like Basic and assembler

Later you will see the Logic/Declarative style

We now turn to the *Procedural* style

Styles

A “style” is an approach to programming that is meant to make it easier for you to write correct programs

Styles

A “style” is an approach to programming that is meant to make it easier for you to write correct programs

It is easy to write small programs in a slapdash manner: you can get away with it as you can hold the whole program in your head

Styles

A “style” is an approach to programming that is meant to make it easier for you to write correct programs

It is easy to write small programs in a slapdash manner: you can get away with it as you can hold the whole program in your head

When projects get large you cannot do this

Styles

A “style” is an approach to programming that is meant to make it easier for you to write correct programs

It is easy to write small programs in a slapdash manner: you can get away with it as you can hold the whole program in your head

When projects get large you cannot do this

Styles are invented to direct the way you write code so to make large systems written by many programmers possible

Styles

They encapsulate detail into simpler blobs to help you keep a grasp on what is happening in your program

Styles

They encapsulate detail into simpler blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Styles

They encapsulate detail into simpler blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Those blobs might be objects, modules, procedures or other things

Styles

They encapsulate detail into simpler blobs to help you keep a grasp on what is happening in your program

You then think “at a higher level” using blobs

Those blobs might be objects, modules, procedures or other things

Roughly speaking, the nature of the blobs is what distinguishes between the various styles

Styles

One thing to remember before we start: programming styles are not exclusive

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

You can write in an OO style in C

Styles

One thing to remember before we start: programming styles are not exclusive

You can write in a procedural style in Java
—though Java makes it hard to do so easily

You can write in an OO style in C
—though C doesn't really provide the constructs for you to do so

Styles

Some languages *support* certain styles

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO

Other languages allow you to use a style, but you have to do the twiddly bits yourself

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO

Other languages allow you to use a style, but you have to do the twiddly bits yourself

You can program OO in C, but you have to do the method lookup yourself

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO

Other languages allow you to use a style, but you have to do the twiddly bits yourself

You can program OO in C, but you have to do the method lookup yourself

Sometimes this is good as it allows you to optimise for the problem in hand

Styles

Some languages *support* certain styles

Java was designed from scratch to be OO

Other languages allow you to use a style, but you have to do the twiddly bits yourself

You can program OO in C, but you have to do the method lookup yourself

Sometimes this is good as it allows you to optimise for the problem in hand

Java provides a general mechanism (that you don’t see) for method lookup that has to work for all kinds of problems

Styles

From the other direction, some *problems* lend themselves better to a certain style

Styles

From the other direction, some *problems* lend themselves better to a certain style

So some problems naturally suggest a choice of language to use to implement them

Styles

From the other direction, some *problems* lend themselves better to a certain style

So some problems naturally suggest a choice of language to use to implement them

Others problems are near impossible to program regardless of style

Styles

From the other direction, some *problems* lend themselves better to a certain style

So some problems naturally suggest a choice of language to use to implement them

Others problems are near impossible to program regardless of style

Part of being a Computer Scientist is knowing these styles and knowing which languages support them

Styles

From the other direction, some *problems* lend themselves better to a certain style

So some problems naturally suggest a choice of language to use to implement them

Others problems are near impossible to program regardless of style

Part of being a Computer Scientist is knowing these styles and knowing which languages support them

And then picking a style for a problem, then a language

Styles

We don’t just try to solve every problem in Java (or C, or . . .)