Imperative Programming with C and C++

Friday 18th January 2008     Time: 09:45 – 11:45

Answer ONE Question from each of the THREE SECTIONS.
Use a SEPARATE answerbook for EACH question

The use of electronic calculators is NOT permitted
Section A
Answer ONE question from this section.

1. a) What do the following terms mean? (5 marks)
   i) Source code
   ii) Assembler code
   iii) Object code
   iv) Linker
   v) Executable code

b) The following function may work correctly and return a copy of the string passed as a parameter. However, there are three potentially serious flaws: what are they? (5 marks)

   ```c
   char strcpy (char *string)
   {
     int i = 0;
     char copy[20];
     while (string[i] != \’\0\’) copy[i] = string[i++];
     return copy;
   }
   ```

c) Write a program to read in 10 real numbers (double) and compute the average, maximum and minimum values. (10 marks)
2. a) Would a definition of the following form potentially cause problems:

```c
#define VALUE 100+MAXLEN
```

If yes, what would be the solution? (2 marks)

b) What do the <> brackets around a filename in a #include directive signify? What would "" in place of the <> mean instead? (2 marks)

c) Write a function named `outOfOrder` that takes as parameters an array of `double` and an `int` parameter named `size` and returns a value of type `int`. The function will test whether the array is out of order by checking if it violates the following condition:

```c
a[0] <= a[1] <= a[2] <= ...
```

The function returns -1 if the elements are not out of order. Otherwise, it will return the index of the first element of the array that is out of order. (6 marks)

d) Write functions that do the following:

i) Compare two strings for equality. If they are equal, zero is returned, otherwise a non-zero value is returned.

ii) Find the first occurrence of a given character in a given string. Return the index to the character's first occurrence in the string, or -1 if it is not found.

iii) Take two strings as arguments. If the first exists in the second as a substring, return the index to the start of the substring's first occurrence in the string, otherwise return -1. (10 marks)
Section B
Answer ONE question from this section.

3. a) What is the relationship, in C, between “pointers” and “arrays”? Illustrate your answer by referring to the following code fragment, and describe, illustrating your answer with a sketch, how the variables would be allocated in memory.

(5 marks)

```c
int myArray[5];
int *myPtr;
int i;

myPtr = myArray;

for (i = 0; i < 5; i++) {
    myArray[i] = i * 2;
}
```

b) Consider the following code:

```c
float myFloatArray[10];
int j;

for (j = 0; j < 11; j++) {
    myFloatArray[j] = j / 2.0;
}
```

i) Explain what you would expect to happen if this code were executed, and describe why it would happen. (3 marks)

ii) If this code were to be translated into Java, and executed, what would you expect to happen, and why? (2 marks)

c) Explain the function of the C operators & and *, taking special care to distinguish between them, and to comment on what data types they can be applied to. (3 marks)

d) Consider the following code:

```c
int k = 23; int m = 37;
int *p1, *p2;
int **p3;

/* A */ p1 = &k;
/* B */ p2 = &m;
/* C */ p3 = &p2;

/* D */ *p1 = 101;
/* E */ (*p2)++;
/* F */ (**p3) --;
/* G */ printf ("%p %p %p \n", p1, p2, p3);
```

Explain, illustrating your answer with diagrams of memory usage, what effects the variable assignments have, in Lines A through F, and what might be printed in Line G. (7 marks)
4. a) Consider the following code:

```c
int fun_a (int a) {
    a++;
}

int fun_b (float *b) {
    (*b) += 0.1;
}

int i = 4;
float f = 3.14;
/* A */ fun_a (i);
/* B */ fun_b (&f);
```

For each of Lines A and B, describe in detail what happens when the functions `fun_a()` and `fun_b()` are called, and what values the variables have after the function calls. Your explanation should include details of the calling mechanism and the role of the stack. (6 marks)

b) What is the difference, in C, between static memory, and dynamic memory? Give a real-world programming example which illustrates why dynamic memory is needed. (4 marks)

c) Imagine you have to write a program which will allow the user to build up a library of images, using a dynamically-managed data structure. The sizes of the images are unknown in advance, as are the number of images the user wishes to store. Each image will comprise the following data: name, size in pixels, array of pixels. While the program is running, the user can specify that a new image is added to the library stored by the program, and will specify the name of a file from which the image may be read. The user may also request that an image be deleted from the program’s data structure.

i) Suggest a conceptual design for a suitable type of data structure to store the images. (2 marks)

ii) Suggest how you might express this design in C (by declaring, for example, a suitable `struct` type). (2 marks)

iii) When the user inputs a new image, suggest how you might add this image to your data structure. (2 marks)

iv) When the user requests deletion of a stored image with a certain name, suggest how you might delete this image from your data structure. (2 marks)

v) How might you make sure there is enough dynamic data space available, before attempting to add a new image? (2 marks)
Section C
Answer ONE question from this section.

5. A C++ class is needed to perform simple computations on a set of calendar dates. Each date is to be represented by an instance of a single class.

a) Give a suitable set of members to store the internal representation of a date, explain why you have chosen this representation and state any assumptions you are making about how the class is to be used.

b) Give a suitable constructor for this class, and explain its value over using a plain C ‘struct’.

c) Explain the purpose of a C++ destructor, and describe what it must do (if anything) in the context of your Date class.

The class requires methods that will enable the date to be converted into a simple printable string, such as “21st September 1970”. It is also necessary to provide for the comparison of dates (e.g. ‘how many days between X and Y’), and to allow dates to be accumulated (such as ‘What is the date 14 days after Z?’).

Explain what ‘operator overloading’ means, and describe how it can be used to provide an intuitive programming interface for each of these operations. Give example signatures for each of the methods.

d) What problems can arise from inappropriate use of operator overloading? Give examples using your Date class.

e) What is ‘name mangling’, and why is it necessary in compiled C++ programs? Give examples of mangled names using your Date class (you may invent your own mangling scheme).

6. a) What is meant by ‘multiple inheritance’ in C++? How does Java provide similar functionality? Explain which of these, in your opinion, is better.

b) What is meant by a ‘polymorphic function’? How are these invoked?

c) How does polymorphism in C++ differ from that in Java? What is the performance implication of this difference?

d) Assume that your C++ compiler translates into plain C as an intermediate step. Explain, using examples and small fragments of C, how code for implementing virtual functions can be generated by the C++ compiler.

e) What is generic programming, and how is this provided for in C++? Give examples. How does Java provide similar functionality? Discuss the pros and cons of these different approaches.