MORE ON STRINGS

Today

- Last time we looked at some basic operations one can carry out on strings
- This time we will look at more complex operations.
- These operations are all illustrated in the program `strings.cpp` which you can download from the course web site.

Member functions

- Most of what we'll cover today is about member functions.
- The idea of member function will make more sense later in the course when we have covered classes.
- But for now, you just have to know that in C++, a string is a *class*, and classes come along with *member functions* or *methods* that operate on them.
- In fact we already met some of these member functions:
  - `cout.precision`
  - `infile.open`

- An obvious thing to find out about a string is how long it is.

  ```
  int    len;
  string dna;

  len = dna.length;
  ```

  will do this for the string `dna`.
- So will:

  ```
  len = dna.size;
  ```

- So far as I can tell, `length` and `size` give exactly the same thing.

- In fact, `len` shouldn't be an `int`.
- We should really use:

```
string::size_type len;
```

- In other words, what gets returned by `size` and `length` is a value of type `string::size_type`.

Finding things in strings

- Often we want to look for things in a string.
- In DNA we typically want to search for short sequences of DNA "letters".
- C++ allows us to do this:

```
string::size_type pos;
pos = dna.find("tata", 0);
```

`pos` gives the location of the start of the first occurence of the string `tata`.

The `0` says to start looking from the first character in `dna`. (Since the string is an array, the first character is numbered 0).

- We can also look for a single character:

```
pos = dna.find('c', 0);
```

- If `dna.find` doesn't find the thing we are looking for, it returns the value `dna.npos`.
- This gives us a neat way to search for things in `dna`.
- We keep looking until we get `dna.npos`.
- So, to count how many times we have g in `dna`, we would do this:

```
int countG = 0;

pos = dna.find('g',0);
while (pos != dna.npos)
{
    countG++;
    pos = dna.find('g', pos + 1);
}
```

- This code works as follows:
  1. We look for g starting at the beginning of the string.
  2. If we don't get `npos` we have found a g, so increase the counter.
  3. Look again, starting with the character just after the one you just found.
  4. Go to 2.
- This is a common way of using a `while` loop.
- We'll see later how to use it to read a file.

Replacing part of a string

- If we want to swap one bit of a string for another, we can use `replace`.
- For example:

  ```
  dna.replace(7, 4, "gaga");
  ```

  will replace the 4 characters that start in place 7 of the string `dna` with the string `gaga`.
- This is fine if you want to swap `gaga` for `tata`, but is no good if you want to take out four characters and put in three, or take out three and put in four.

- To swap two bits of a string that aren't the same length, we have to first `erase` one and then `insert` another.
- For example:

  ```
  dna.erase(7, 4);
  dna.insert(7, "ctctc");
  ```

  will remove the four characters of `dna` that start with the character in place number seven, and then insert the string `ctctc` at the same place.

- A slightly more sophisticated use of `insert` and `erase` is:

  ```
  pos = dna.find("ggaa", 0);
  dna.erase(pos, 4);
  dna.insert(pos, "tatatt");
  ```

- This finds the location of the first string `ggaa`, `erase`s four characters at that position, and then `insert`s `tatatt` in the same place.
- The overall effect is to replace `ggaa` with `tatatt`

Extracting part of a string

- If we want to grab a bit from the middle of a string, we can use `substr`.
- This extracts a *substring* from the string we apply it to.
- For example:

  ```
  string dnaPart;
  dnaPart = dna.substr(7, 4);
  ```

  will copy the 4 characters that start in place 7 of the string `dna` into the string `dnaPart`.

Two other things

- Just as we can concatenate two strings using +

  ```
  dna = dna + dna2;
  ```

  we can combine concantenation and assignment using +=

  ```
  dna += dna2;
  ```

- `dna.erase()` will set `dna` to contain no characters.
- This is the same as doing:

  ```
  dna = "";
  ```

Summary

- This lecture looked at a number of the member functions of the `string` class:
  - `length`
  - `size`
  - `find`
  - `npos`
  - `replace`
  - `erase`
  - `insert`
  - `substr`
  - `clear`
- We will talk more of strings in the next lecture.