Exercises 1

张逸松

September 1, 2019

1.3

My functions

```
// Copy s1 to s2
ADT_String strcpy(ADT_String s1, ADT_String s2);

// Concatenate two strings
ADT_String strcat(ADT_String s1, ADT_String s2);

// Return the length of s
int strlen(ADT_String s);

// Intercept the substring from s, starting at 'pos', and of length of '
    size'
ADT_String substr(ADT_String s, int pos, int size);

// Return 0 (if s1 is equal to s2), 1 (if s1 is greater then s2), -1 (if
     s1 is less than s2)
int strcmp(ADT_String s1, ADT_String s2);

// Find the location where c first appeared in s (0-based)
int strchr(ADT_String s, char c);

// Reverse s
ADT_String strrev(ADT_String s);
```

First physical representation

```
class ADT_String
{
    vector <char> s;
};
```

Second physical representation

```
class ADT_String
{
    char c;
    ADT_String *nxt;
};
```

1.4

An ADT for a list of integers might specify the following operations:

- Insert a new integer at a particular position in the list.
- Return **true** if the list is empty.
- Reinitialize the list.
- Return the number of integers currently in the list.
- Delete the integer at a particular position in the list.

Code

```
class ADT_List_Integer
{
private:
    int value;
    ADT_List_integer* nxt;
protected:
    virturl void push_back(int x);
    virturl bool empty();
    virturl void clear();
    virturl int size();
    virturl void delete();
};
```

1.6

```
ADT add(ADT a, ADT b);
ADT multiply(ADT a, ADT b);
ADT transpose(ADT a);
void setValue(ADT a, int x, int y, int value);
```

```
int getValue(ADT a, int x, int y);
```

Implementation

One implementation is a two-dimensional hash table, another one is the adjacency table.

1.7

```
funcion BublleSort(A)
    n <- the length of A
    k <- n - 1
        for step = 1 to k do
            for i = 2 to n do
                if A[i - 1] > A[i] then
                    swap A[i - 1] and A[i]
```

1.8

```
map <ADT1, ADT2> mp;
ADT1 key; ADT2 value;
auto it = mp.find(key);
it->second = value;
```