HOMEWORK 12 REPORT

Overview

The `CustomDict` library provides a dynamic dictionary-like data structure for the C programming language, allowing storage and manipulation of key-value pairs where keys are strings and values can be of different types (`int`, `float`, `double`, `char`). This library includes functionality for creating dictionaries, adding and deleting items, updating values, searching for items, sorting the dictionary, printing its contents, and reading data from CSV files.

Data Structures

Value Union:
```
typedef union Value {
   int i;
   float f;
   double d;
   char c;
} Value;
```

The Value union allows a variable to store values of different types (integer, float, double, and character), facilitating flexibility in the type of data stored in the dictionary.

Item Structure:
```
typedef struct Item {
   char *key;
   Value value;
   char type;
} Item;
```

The Item structure represents a key-value pair:
- key is a dynamically allocated string.
- value is a Value union holding the item's value.
- type is a character indicating the type of the value (‘i’ for integer, ‘f’ for float, ‘d’ for double, and ‘c’ for char).

CustomDict Structure
```
typedef struct CustomDict {
   Item *items;
   int capacity;
   int size;
} CustomDict;
```

The CustomDict structure represents the entire dictionary:
- items is a dynamically allocated array of Item structures.

- capacity indicates the allocated size of the items array.
- size is the current number of items in the dictionary.

Function Descriptions

--create_dict—
struct CustomDict* create_dict(void);

Description:
This function creates and initializes a new `CustomDict` instance with zero items and returns a pointer to it.

Usage:
CustomDict* myDict = create_dict();

Returns:
A pointer to the newly created `CustomDict` structure.

--add_item--
void add_item(struct CustomDict* dict, char* key, union Value* value, char type);

Description:
Adds a new item to the dictionary or updates the value of an existing item if the key already exists.

Parameters:
- dict: A pointer to the `CustomDict` structure.
- key: A string representing the key of the item.
- value: A pointer to a `Value` union holding the value to be added.
- type: A character representing the type of the value.

Details:
- Checks if the dictionary's capacity needs to be increased.
- Searches for the key; if found, updates the existing value.
- If the key is not found, adds a new item with the specified key and value.

Usage:
Value val;
val.i = 42;
add_item(myDict, "exampleKey", &val, 'i');

--delete_item--
void delete_item(struct CustomDict* dict, char* key);

Description:
Removes an item from the dictionary based on its key.

Parameters:
- dict: A pointer to the `CustomDict` structure.
- key: A string representing the key of the item to be deleted.

Details:
- Searches for the key in the dictionary.
- If the key is found, removes the item and shifts the remaining items to fill the gap.

Usage:
delete_item(myDict, "exampleKey");

--set_value--
void set_value(struct CustomDict* dict, char* key, union Value* value);

Description:
Updates the value of an existing item in the dictionary based on its key.

Parameters:
- dict: A pointer to the `CustomDict` structure.
- key: A string representing the key of the item.
- value: A pointer to a `Value` union holding the new value.

Details:
- Searches for the key in the dictionary.
- If the key is found, updates the value.

Usage:
Value newVal;
newVal.f = 3.14;
set_value(myDict, "exampleKey", &newVal);

--search_item--
union Value* search_item(struct CustomDict* dict, char* key);

Description:
Searches for an item in the dictionary by its key and returns a pointer to its value.

Parameters:
- dict: A pointer to the `CustomDict` structure.
- key: A string representing the key of the item to be searched for.

Returns:
A pointer to the `Value` union of the found item, or `NULL` if the item is not found.

Usage:

```
Value* foundVal = search_item(myDict, "exampleKey");
if (foundVal != NULL) {
   printf("Found value: %d\n", foundVal->i);
}
```

--sort_dict--
```
void sort_dict(struct CustomDict* dict);
```

Description:
Sorts the items in the dictionary alphabetically by their keys.

Parameters:
- dict: A pointer to the `CustomDict` structure.

Details:
- Uses a simple bubble sort algorithm to sort the items.

Usage:
```
sort_dict(myDict);
```

--print_dict--
```
void print_dict(struct CustomDict* dict);
```

Description:
Prints all the items in the dictionary to the standard output.

Parameters:
- dict: A pointer to the `CustomDict` structure.

Details:
- Iterates through the dictionary items and prints each key and value based on its type.

Usage:
```
print_dict(myDict);
```

--free_dict--
```
void free_dict(struct CustomDict* dict);
```

Description:
Frees all the allocated memory associated with the dictionary.

Parameters:
- dict: A pointer to the `CustomDict` structure.

Details:

- Frees the memory allocated for each key and the `items` array.
- Frees the `CustomDict` structure itself.

Usage:
free_dict(myDict);

--read_csv_find_size--
void read_csv_find_size(int file_info[100], const char* filename, int* total_size);

Description:
Reads a CSV file to determine the number of columns and rows, storing this information in an array.

Parameters:
- file_info: An array of integers to store the number of columns in each row.
- filename: The name of the CSV file.
- total_size: A pointer to an integer to store the total number of rows.

Details:
- Opens the CSV file and reads it character by character.
- Counts the number of columns (comma-separated values) in each row.
- Stores the count in `file_info` and the number of rows in `total_size`.

Usage:
int file_info[100];
int total_size;
read_csv_find_size(file_info, "data.csv", &total_size);

--read_csv--
int read_csv(int file_info[100], CustomDict *dicts, const char* filename, const int size);

Description:
Reads data from a CSV file and populates a dictionary with the parsed key-value pairs.

Parameters:
- file_info: An array of integers representing the number of columns in each row.
- dicts: A pointer to the `CustomDict` structure to be populated.
- filename: The name of the CSV file.
- size: The total number of rows in the CSV file.

Returns:
An integer indicating success (‘1’) or failure (‘0’).

Details:
- Opens the CSV file and reads it character by character.
- Parses each row based on the column count specified in file_info.

- Adds the parsed key-value pairs to the dictionary.

Usage:

```
CustomDict* myDict = create_dict();
int file_info[100];
int total_size;
read_csv_find_size(file_info, "data.csv", &total_size);
if (read_csv(file_info, myDict, "data.csv", total_size)) {
   printf("CSV data loaded successfully\n");
}
```

The `CustomDict` library provides a versatile and dynamic way to manage key-value pairs in C, with support for multiple value types and basic operations such as adding, deleting, updating, searching, and sorting items. Additionally, the library includes functions to read and parse data from CSV files, making it useful for applications requiring dynamic data management.

Below are some outputs of the code:

Main:

```c
int main(void) {
    struct CustomDict* dict = create_dict();
    int totalsize ;
    int file_info[100];



    read_csv_find_size(file_info, File_Path, &totalsize);
    read_csv(file_info, dict, File_Path,totalsize);
    print_dict(dict);
    union Value* search_result = search_item(dict, "count2");
    if (search_result) {
            printf("Found value for 'count2': %d\n", search_result->i);
        } else {
            printf("Key not found\n");
        }
    printf("------------\n");
    sort_dict(dict);
    print_dict(dict);

    delete_item(dict, "size3");
    printf("------------\n");
    print_dict(dict);
    Value val1;
    val1.i = 111;
    set_value(dict, "size2", &val1);
    print_dict(dict);

    return 0;
}
```

Then the result:

First dict view:

```
Key: age0  25
Key: age1  12
Key: age2  556
Key: age3  1
Key: age4  192561
Key: weight0  12.560000
Key: weight1  664.200000
Key: weight2  5.500000
Key: blood_type0  A
Key: blood_type1  B
Key: blood_type2  C
Key: blood_type3  D
Key: blood_type4  E
Key: blood_type5  F
Key: height0  10.500000
Key: height1  20.250000
Key: height2  30.750000
Key: height3  40.200001
Key: score0  30
Key: score1  45
Key: score2  120
Key: price0  9.800000
Key: price1  7.600000
Key: blood_typ0  X
Key: blood_typ1  Y
Key: blood_typ2  Z
Key: value0  15.300000
Key: value1  25.799999
Key: value2  35.599998
Key: value3  48.900002
Key: value4  56.200001
Key: quantity0  5
Key: quantity1  10
Key: quantity2  15
Key: quantity3  20
Key: amount0  18.900000
Key: amount1  24.700000
Key: amount2  33.600000
Key: letter0  M
Key: letter1  N
Key: letter2  O
Key: letter3  P
Key: temperature0  22.500000
Key: temperature1  30.799999
Key: temperature2  38.099998
Key: temperature3  42.599998
Key: count0  8
Key: count1  16
Key: count2  32
Key: distance0  55.400000
Key: distance1  67.900000
Key: speed0  40.500000
Key: speed1  55.200001
Key: speed2  60.900002
Key: speed3  75.300003
Key: size0  100
Key: size1  200
Key: size2  300
Key: size3  400
Key: balance0  500.250000
Key: balance1  1000.500000
61 item was founded
Found value for 'count2': 32
```

After sort the dict:

```
Found value for 'count2': 32
-----------
Key: age0  25
Key: age1  12
Key: age2  556
Key: age3  1
Key: age4  192561
Key: amount0  18.900000
Key: amount1  24.700000
Key: amount2  33.600000
Key: balance0  500.250000
Key: balance1  1000.500000
Key: blood_typ0  X
Key: blood_typ1  Y
Key: blood_typ2  Z
Key: blood_type0  A
Key: blood_type1  B
Key: blood_type2  C
Key: blood_type3  D
Key: blood_type4  E
Key: blood_type5  F
Key: count0  8
Key: count1  16
Key: count2  32
Key: distance0  55.400000
Key: distance1  67.900000
Key: height0  10.500000
Key: height1  20.250000
Key: height2  30.750000
Key: height3  40.200001
Key: letter0  M
Key: letter1  N
Key: letter2  O
Key: letter3  P
Key: price0  9.800000
Key: price1  7.600000
Key: quantity0  5
Key: quantity1  10
Key: quantity2  15
Key: quantity3  20
Key: score0  30
Key: score1  45
Key: score2  120
Key: size0  100
Key: size1  200
Key: size2  300
Key: size3  400
Key: speed0  40.500000
Key: speed1  55.200001
Key: speed2  60.900002
Key: speed3  75.300003
Key: temperature0  22.500000
Key: temperature1  30.799999
Key: temperature2  38.099998
Key: temperature3  42.599998
Key: value0  15.300000
Key: value1  25.799999
Key: value2  35.599998
Key: value3  48.900002
Key: value4  56.200001
Key: weight0  12.560000
Key: weight1  664.200000
Key: weight2  5.500000
61 item was founded
```

After delete the ‘size3’

```
============
Key: age0  25
Key: age1  12
Key: age2  556
Key: age3  1
Key: age4  192561
Key: amount0  18.900000
Key: amount1  24.700000
Key: amount2  33.600000
Key: balance0  500.250000
Key: balance1  1000.500000
Key: blood_typ0  X
Key: blood_typ1  Y
Key: blood_typ2  Z
Key: blood_type0  A
Key: blood_type1  B
Key: blood_type2  C
Key: blood_type3  D
Key: blood_type4  E
Key: blood_type5  F
Key: count0  8
Key: count1  16
Key: count2  32
Key: distance0  55.400000
Key: distance1  67.900000
Key: height0  10.500000
Key: height1  20.250000
Key: height2  30.750000
Key: height3  40.200001
Key: letter0  M
Key: letter1  N
Key: letter2  O
Key: letter3  P
Key: price0  9.800000
Key: price1  7.600000
Key: quantity0  5
Key: quantity1  10
Key: quantity2  15
Key: quantity3  20
Key: score0  30
Key: score1  45
Key: score2  120
Key: size0  100
Key: size1  200
Key: size2  300
Key: speed0  40.500000
Key: speed1  55.200001
Key: speed2  60.900002
Key: speed3  75.300003
Key: temperature0  22.500000
Key: temperature1  30.799999
Key: temperature2  38.099998
Key: temperature3  42.599998
Key: value0  15.300000
Key: value1  25.799999
Key: value2  35.599998
Key: value3  48.900002
Key: value4  56.200001
Key: weight0  12.560000
Key: weight1  664.200000
Key: weight2  5.500000
60 item was founded
```

After change the size2 value:

```
Key: age0  25
Key: age1  12
Key: age2  556
Key: age3  1
Key: age4  192561
Key: amount0  18.900000
Key: amount1  24.700000
Key: amount2  33.600000
Key: balance0  500.250000
Key: balance1  1000.500000
Key: blood_typ0  X
Key: blood_typ1  Y
Key: blood_typ2  Z
Key: blood_type0  A
Key: blood_type1  B
Key: blood_type2  C
Key: blood_type3  D
Key: blood_type4  E
Key: blood_type5  F
Key: count0  8
Key: count1  16
Key: count2  32
Key: distance0  55.400000
Key: distance1  67.900000
Key: height0  10.500000
Key: height1  20.250000
Key: height2  30.750000
Key: height3  40.200001
Key: letter0  M
Key: letter1  N
Key: letter2  O
Key: letter3  P
Key: price0  9.800000
Key: price1  7.600000
Key: quantity0  5
Key: quantity1  10
Key: quantity2  15
Key: quantity3  20
Key: score0  30
Key: score1  45
Key: score2  120
Key: size0  100
Key: size1  200
Key: size2  111
Key: speed0  40.500000
Key: speed1  55.200001
Key: speed2  60.900002
Key: speed3  75.300003
Key: temperature0  22.500000
Key: temperature1  30.799999
Key: temperature2  38.099998
Key: temperature3  42.599998
Key: value0  15.300000
Key: value1  25.799999
Key: value2  35.599998
Key: value3  48.900002
Key: value4  56.200001
Key: weight0  12.560000
Key: weight1  664.200000
Key: weight2  5.500000
60 item was founded
Program ended with exit code: 0
```