2021-2022 Cprog_Homework

Author: H3Art

HW1——Calculates the distance

Write, compile， and execute a C program that calculates the distance that light travels in one year, called a light-year. Given that light travels at a speed of 300000000 meters in one second, determine the distance of a light-year, The relevant formula is distance=speed* time. (Hint: Make sure that the time corresponds to the number of seconds in one year)

```c
#include <stdio.h>

int main() {
    long long speed = 300000000;
    long long time = 60 * 60 * 24 * 365;
    long long distance = speed * time;
    printf("%lld", distance);
    return 0;
}
```

HW2——Estimate population & Calculate the amount of money

Solve the following two problems in one single C program:

1. A model of world population, in billions of people, after 2000 is given by the equation Population = $6.0e^{.02[Year-2000]}$. Using this formula, estimate the world population in the year 2010.

2. Calculate and display the amount of money, A, available in N years when an initial deposit of X dollars is deposited in a bank account paying an annual interest rate of R percent. Use the relationship that

A=X(1.0 + R/100)N.

```c
#include <math.h>
#include <stdio.h>

int main() {
    double p = 6.0 * exp(0.02 * (2010 - 2000));
    double A, R, X, N;
    /*R为利率，X为本金,N为存的时间（年）*/
    printf("%lf\n", p);
    scanf("%lf %lf %lf", &R, &X, &N);
    A = X * pow((1.0 + R / 100), N);
    printf("%lf", A);
    return 0;
}
```

HW3——Switch Statements

Input two floating point numbers num1, num2 with an operator op(+, -, *, /) from the keyboard, use switch statements to select and perform the corresponding calculation, for example: input "23.0+37.0", then output "23.00+37.00=60.00"; input "2.0*3.0", then output "2.00*3.00=6.00".

```c
#include <stdio.h>

int main() {
    float num1, num2;
    char op;
    scanf("%f%c%f", &num1, &op, &num2);
    switch (op) {
        case '+':
            printf("%.2f + %.2f = %.2f", num1, num2, num1 + num2);
            break;
        case '-':
            printf("%.2f - %.2f = %.2f", num1, num2, num1 - num2);
            break;
        case '*':
            printf("%.2f * %.2f = %.2f", num1, num2, num1 * num2);
            break;
        case '/':
            printf("%.2f / %.2f = %.2f", num1, num2, num1 / num2);
            break;
    }
    return 0;
}
```

HW4——Nested Loops

Use nested loops to print out a hollow pyramid shape with any number of rows:

```
Eg: numRows  = 5
    *
   * *
  *   *
 *     *
*********
```

```c
#include <stdio.h>

int main() {
    int row = 0;
    scanf("%d", &row);
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < row - i - 1; j++) {
            printf(" ");
            if (j == (row - i - 2)) {
                printf("*");
            }
        }
        if (i != (row - 1)) {
            for (int k = 0; k < 2 * i - 1; k++) {
                printf(" ");
                if (k == (2 * i - 2)) {
                    printf("*");
                }
            }
        } else {
            for (int k = 0; k < 2 * i + 1; k++) {
                printf("*");
            }
        }
        printf("\n");
    }
    return 0;
}
```

HW5——Random Number

1. Write a C program to create a HI-LO game. In this game, the computer produces a random integer between 1 and 100 and provides the user with seven tries to guess the generated number. If the user guesses the correct number, the message, “Hooray, you have won!” should be displayed. After each incorrect guess, the computer should display the message, “Wrong Number, Try Again” and indicate whether the guess was too high or too low and display the number of guesses left. After seven incorrect guesses, the computer should display the message, “Sorry, you lose” and the correct number.
2. Modify the program written for Exercise 6a to allow the user to run the game again after a game has been completed. The program should display the message “Would you like to play again (y/n) ? ” and

restart if the user enters either a Y or y.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    srand((unsigned)time(NULL));
    int in;
    c
    while (1) {
        int random = rand() % 100 + 1;

        for (int i = 0; i < 7; i++) {
            printf("Please guess a number between 1 and 100:\n");
            scanf("%d", &in);

            if (in > random) {
                printf("Wrong Number, Try Again.\nThe guess is too
high.\n");
            } else if (in < random) {
                printf("Wrong Number, Try Again.\nThe guess is too
low.\n");
            } else {
                printf("Hooray, you have won!\n");
                break;
            }

            printf("The number of guesses left is %d.\n", 6 - i);

            if (i == 6) {
                printf("\nSorry, you lose,the correct number is %d.",
random);
            }

            printf("\n");
        }

        printf("Would you like to play again (y/n) ?\n");
        scanf("\n%c", &re);

        if (re == 'Y' || re == 'y') {
            printf("\n");
            continue;
        } else {
            break;
        }
    }
    // system("PAUSE");
    return 0;
}
```

HW6——Recursion

Write a C program to calculate e^x, the x-th power of e, using the following formula:

$e^x \approx 1 + x + x^2/2! + x^3/3! + \ldots + x^{19}/19!$

Write the program with three functions by using recursions:

float fun1(int x, int n) // calculate x^n, No using loops or math.h pow() function.

float fun2(int n) // calculate n!, No using loops.

float fune(int x, int n) // calculate e^x , where n=19, No using loops.

```c
#include <stdio.h>

// fun1是幂函数
float fun1(int x, int n) {
    if (n > 1) {
        return x * fun1(x, n - 1);
        //传入的n若大于0，不断递至n为1，恰好满足幂次数为n次
    } else if (n == 0) {
        return 1;  //传入的n若为0，直接x^0 = 1
    } else {
        return x;  // n为1时，归
    }
}

// fun2是阶乘函数
float fun2(int n) {
    if (n > 1) {
        return n * fun2(n - 1);
        //传入的n若大于0，不断递至n为1，恰好满足n阶乘
    } else if (n == 0) {
        return 1;  //传入的n若为0，规定0! = 1
    } else {
        return n;  //当n为1时，归
    }
}

// fune是e^x的泰勒公式
float fune(int x, int n) {
    float sum = fun1(x, n) / fun2(n);  //从最大阶数开始
    if (n > 0) {
        return fune(x, n - 1) + sum;  //不断递向低阶数，直至n = 1
    } else {
        return sum;
        //n = 0时，此时当前层函数的sum值为1，开始归
    }
}

int main(void) {
    int x = 0, n = 0;
```

```
    scanf("%d%d", &x, &n);
    float output = fune(x, n);
    printf("%f", output);
    return 0;
}
```

HW7——Array & Sorting

An 1-D array has ten elements: 4.4 3.3 2.2 5.5 1.1 6.6 7.7 10.0 9.9 8.8.

Write a program to sort the array using two different sorting methods (selection sort, bubble sort or quicksort).

Each sorting function provides an extra parameter: a flag character for deciding whether to sort in descending or ascending order ('a' for ascending; 'd' for descending). e.g. Bubble sort in descending order: bubble_sort(a2, n, 'd').

It is highly recommended that you make full use of the example program code in Chapter 8.

Example program code: Csource.rar

```
#include <stdio.h>

void swap(double* num1, double* num2)
{
    double temp = *num1;
    *num1 = *num2;
    *num2 = temp;
    return;
}

//冒泡排序
void bubble_sort(double* arr, int len, char sequence)
{
    //传入数组名会退化为数组的首地址，但依然可以在函数内用数组的方式操作整个数组变量
    //冒泡排序的思路是每次让指向的元素与下一个元素相比较，满足比较条件就进行交换
    //这样做最终会在每一轮比较后将最大/最小的元素移至比较范围内数组的一端
    double temp; //定义临时变量以进行后面步骤的值调换
    for (int i = 0; i < len - 1; i++) {
        //外循环代表冒泡排序总共需要进行（数组元素个数-1）次
        for (int j = 0; j < len - 1 - i; j++) {
            //内循环代表某一次排序过程中要进行（剩余未排序的数组元素个数-1）次
            switch (sequence) {
            case 'a':
                if (arr[j] > arr[j + 1]) {
                    swap(&arr[j], &arr[j + 1]);
                }
                break;
            case 'd':
                if (arr[j] < arr[j + 1]) {
                    swap(&arr[j], &arr[j + 1]);
```

```
                }
                break;
            }
        }
    }
}

//选择排序
void selection_sort(double* arr, int len, char sequence)
{
    //选择排序在每一轮中选出最大/最小的元素，然后将其移至选择范围内数组的一端
    double max, min, temp;
    int rec;
    switch (sequence) {
    case 'a':
        for (int i = 0; i < len - 1; i++) {
            //总共需要选择（数组元素个数-1）次
            min = arr[i];
            //每次开始的时候将最大值/最小值赋值为第一个元素
            rec = i;
            //每次开始的时候需要初始化一个标记，标记出最终为最大/最小值的数组下标
            for (int j = i + 1; j < len; j++) {
                if (arr[j] < min) {
                    min = arr[j];
                    rec = j;
                }
            }
            //选出最大/最小值后就将选择范围内的第一个元素赋值为最大/最小值
            //同时将标记的那个数组下标对应的元素赋值为未赋值最大/最小值之前的第一个元素
            swap(&arr[rec], &arr[i]);
        }
        break;
    case 'd':
        for (int i = 0; i < len - 1; i++) {
            max = arr[i];
            rec = i;
            for (int j = i + 1; j < len; j++) {
                if (arr[j] > max) {
                    max = arr[j];
                    rec = j;
                }
            }
            swap(&arr[rec], &arr[i]);
        }
        break;
    }
}

//快速排序
void quick_sort(double* arr, int beg, int end, char sequence)
{
    //快速排序利用二分的思想进行排序
    if (beg > end) {
        return;
```

```
    }
    double std = arr[beg];
    //先将数组的第一个元素设为基准
    int i = beg, j = end;
    while (i != j) {
        switch (sequence) {
        case 'a':
            //以升序（正序）排序为例，设置两个哨兵i与j
            //让查找先从j开始，j对应比基准更大的数时直接自减
            while (arr[j] >= std && i < j) {
                j--;
            }
            //同理i对应比基准更小的数时就自增，最终i会指向比基准更大的数
            //j指向比基准更小的数
            while (arr[i] <= std && i < j) {
                i++;
            }
            break;
        case 'd':
            while (arr[j] <= std && i < j) {
                j--;
            }
            while (arr[i] >= std && i < j) {
                i++;
            }
            break;
        }
        //将i和j分别对应的数组元素交换
        if (i < j) {
            swap(&arr[i], &arr[j]);
        }
        //在结束的时候i==j,指向同一个元素
    }
    //最后将数组的第一个元素赋值为i和j指向的那个元素
    arr[beg] = arr[i];
    //然后i和j共同指向的元素赋值为基准值，这样就完成了将基准值前后的值都分成
    //比基准更大和比基准更小的部分
    arr[i] = std;
    //接下来递归重复调用，将每个部分逐渐二分进行快排，完成整个排序
    quick_sort(arr, beg, i - 1, sequence);
    quick_sort(arr, i + 1, end, sequence);
}

int main()
{
    double arr[10] = { 4.4, 3.3, 2.2, 5.5, 1.1, 6.6, 7.7, 10.0, 9.9, 8.8
};
    int sortway = 0;
    char order;
    printf("输入排序的顺序（正序为a倒序为d），再输入排序的方式（冒泡1/选择2/快速3）。
\n");
    scanf("%c%d", &order, &sortway);
    switch (sortway) {
    case 1:
```

```
            bubble_sort(arr, 10, order);
            break;
        case 2:
            selection_sort(arr, 10, order);
            break;
        case 3:
            quick_sort(arr, 0, 9, order);
            break;
        }
        for (int i = 0; i < 10; i++) {
            printf("%.1lf ", arr[i]);
        }
        return 0;
    }
```

HW8——Character & String

Write a C function named isvalidReal () that checks for a valid double-precision number. This kind of number can have an optional + or - sign, at most one decimal point, which can also be the first character and at least one digit between 0 and 9 inclusive. The function should return an integer value of 1 if the entered number is a real number; otherwise, it should return an integer value of 0.

Testing cases:

true: 1) "+900.1" 2) "-900.1" 3) "+.001" 4) ".000" 5) "900." 6) "-999" 7) "999"

false: 1)"" 2) " " 3) "+" 4) "-" 5) "+." 6) "." 7) "99+" 8) "99a0" 9) "900..01" 10) " +.009"

```
#include <stdio.h>
#include <string.h>
#define LEN 100

// 1.
// 若使用scanf，利用%c来接收空格与换行，但使用gets更简单，gets读取到换行直接停止
//    （但似乎gets在stdio.h中已经被删除了？）
// 2. 给出数据的第一个字符只能是+、-、小数点或者数字，而不能是空格或者直接\0
// 3. 除第一个字符以外，剩下的字符只能是数字或小数点
// 4. 小数点出现的次数至多只能有一次
// 5. 至少出现一个0-9的数字

int isvalidReal(char* str) {
    int i = 0, numcnt = 0, dotcnt = 0;
    //第一个字符的判断用一个if单独进行
    if (str[0] != '+' && str[0] != '-' && str[0] != '.' &&
        (str[0] < 48 || str[0] > 57)) {
        return 0;
    }
    //接下去字符判断连续进行直到遇见'\0',但必须从第一个字符开始
    //因为要满足小数点的计数器记录下第一位的小数点，否则“..009”会被判定为有效
    while (str[i] != '\0') {
        if ((str[i] < 48 || str[i] > 57) && str[i] != '.') {
```

```
            return 0;
        } else if (str[i] > 47 && str[i] < 58) {
            numcnt++;
        } else {
            dotcnt++;
        }
        if (dotcnt > 1) {
            return 0;
        }
        i++;
    }
    if (numcnt < 1) {
        return 0;
    } else {
        return 1;
    }
}

int main() {
    char str[LEN];
    memset(str, 0, sizeof(char) * LEN);
    printf(
        "Input the number and finally use \"ENTER\" key to end the
input.\n");
    //利用scanf的解决方案:
    // for (int i = 0; i < LEN; i++) {
    //    scanf("%c", &str[i]);
    //    if (str[i] == '\n') {
    //      str[i] = '\0';
    //      break;
    //  }
    //}
    gets(str);
    int result = isvalidReal(str);
    printf("Your input is \"%s\".\nThe result is %d.\n", str, result);
    if (result) {
        printf("Thus, your input is a valid real number.");
    } else {
        printf("Thus, your input isn't a valid real number.");
    }
    return 0;
}
```

HW9——Files Operation

a. Write, compile, and run a C program that creates a binary file named grades. bin and writes the following five lines of data to the file:

```
90.3    92.7    90.3    99.8
85.3    90.5    87.3    90.8
93.2    88.4    93.8    75.6
```

```
82.4    95.6    78.2    90.0
93.5    80.2    92.9    94.4
```

```c
#include <stdio.h>
#include <stdlib.h>

// Task a):编写出可以将给定成绩表数据写入一个名为grade.bin的二进制文件中
//         利用FILE文件指针配合fopen函数以写模式打开文件
//         再利用fwrite函数写入数据
double num[20] = {90.3, 92.7, 90.3, 99.8, 85.3, 90.5, 87.3, 90.8, 93.2,
88.4,
                  93.8, 75.6, 82.4, 95.6, 78.2, 90.0, 93.5, 80.2, 92.9,
94.4};

int main() {
    FILE* fp;
    //注意判断文件是否正确打开
    //利用“wb”(写)模式打开当前目录下的grade.bin文件，若没有则创建之
    //不能使用fputs函数进行写入，fputs函数只能写入字符型数据而非二进制数据
    //利用fwrite函数可以写入二进制数据，fread可以读取二进制数据
    if ((fp = fopen("./grade.bin", "wb")) != NULL) {
        for (int i = 0; i < 20; i++) {
            fwrite(num + i, sizeof(double), 1, fp);
        }
    } else {
        printf("The file is not open correctly.\n");
        printf("Please check that the file exists\n");
        //若文件没有正确地打开，直接退出程序，返回代码-1表明程序错误结束
        exit(-1);
    }
    //读写结束完毕手动关闭文件流
    fclose(fp);
    printf("Now we have already put the data into the file,\n");
    return 0;
}
```

b. Using the data in the grades. bin file created in Exercise 5a, write, compile, and run a C program that reads, computes and displays the average of each group of four grades. (Hint: use a 2-D array to store the grades)

```c
#include <stdio.h>
#include <stdlib.h>

// Task b):让程序可以读取文件中的数据，数据是五行四列的成绩表
//         因此，使用fread函数扫描文件中的数据
//         最终将数据存储在二维数组中进行平均成绩计算
int main() {
    FILE* fp;
    double grade_table[5][4], avg = 0;
```

```
    //利用“rb”(读)模式打开当前目录下的grade.bin文件
    if ((fp = fopen("./grade.bin", "rb")) != NULL) {
        for (int i = 0; i < 5; i++) {
            avg = 0;
            for (int j = 0; j < 4; j++) {
                //该函数可以读取FILE指针对应文件的内容
                fread(&grade_table[i][j], sizeof(double), 1, fp);
                avg += grade_table[i][j];
            }
            printf("%lf\n", avg / 4);
        }
    } else {
        printf("The file is not open correctly.\n");
        //用非0返回值代表程序运行发生了错误，便于查错
        exit(1);
    }
    fclose(fp);
    return 0;
}
```

HW10——Function Pointer

Rewrite a sorting function, selectionSort() or bubbleSort() :

1. using pointer notations for arrays;
2. adding a pointer to functions as its parameter, dec() or inc() which enables sorting in descending or ascending order respectively;
3. using a 1-D array with ten elements: 4.4 3.3 2.2 5.5 1.1 6.6 7.7 10.0 9.9 8.8 for testing.

```
#include <stdio.h>
#define LEN 10

//目标：1.
//用指针代替数组下标进行选择或冒泡排序(对指针+偏移量的和取值可实现与访问数组对应下标同样的效果)
//      2. 分别写出升序排序以及降序排序

double array[LEN] = {4.4, 3.3, 2.2, 5.5, 1.1, 6.6, 7.7, 10.0, 9.9, 8.8};

//两个变量值交换的函数声明
void Swap_double(double*, double*);
//创建一个函数指针
void (*sort)(double*);

void dec(double* arr) {
    for (int i = 0; i < LEN - 1; i++) {
        for (int j = 0; j < LEN - 1 - i; j++) {
            if (*(arr + j) < *(arr + j + 1)) {
                Swap_double(arr + j, arr + j + 1);
            }
        }
```

```
    }
    return;
}

//利用选择排序进行升序排列操作
void inc(double* arr) {
    for (int i = 0; i < LEN - 1; i++) {
        double min = *(arr + i);
        int rec = i;
        for (int j = i + 1; j < LEN; j++) {
            if (*(arr + j) < min) {
                min = *(arr + j);
                rec = j;
            }
        }
        Swap_double(arr + i, arr + rec);
    }
    return;
}

//封装交换函数
void Swap_double(double* a, double* b) {
    double temp = *a;
    *a = *b;
    *b = temp;
    return;
}

int main() {
    //将升序排序函数的地址赋给一开始创建的函数指针
    sort = &inc;
    //对函数指针取值便可以调用升序排序函数
    (*sort)(array);
    printf(
        "The result by using selection way to sort the array in increasing
"
        "sequence shows as follow:\n");
    for (int i = 0; i < LEN; i++) {
        printf("%.1lf ", *(array + i));
    }
    printf("\n");
    //把函数指针指向的函数地址更改为降序排序函数的地址
    sort = &dec;
    (*sort)(array);
    printf(
        "The result by using bubblesort way to sort the array in
decreasing "
        "sequence shows as follow:\n");
    for (int i = 0; i < LEN; i++) {
        printf("%.1lf ", *(array + i));
    }
    printf("\n");
    return 0;
}
```

HW11——Structure

Person is a structure that includes a name and an age. Assume that We have an array of the structure Person, now please write a program to:

1. sort the array by ages;
2. sort the array by names. (Optional)

```c
#include <stdio.h>
#include <string.h>

#define NUM 10

typedef struct data {
    char name[20];
    int age;
} data;

void sort_name(data* pers, int len) {
    // BubbleSort
    for (int i = 0; i < len - 1; i++) {
        for (int j = 0; j < len - 1 - i; j++) {
            if (strcmp(pers[j].name, pers[j + 1].name) > 0) {
                data temp = pers[j];
                pers[j] = pers[j + 1];
                pers[j + 1] = temp;
            }
        }
    }
    return;
}

void sort_age(data* pers, int len) {
    // SelectionSort
    for (int i = 0; i < len - 1; i++) {
        int maxage = pers[i].age, rec_index = i;
        for (int j = i + 1; j < len; j++) {
            if (pers[j].age < maxage) {
                maxage = pers[j].age;
                rec_index = j;
            }
        }
        data temp = pers[i];
        pers[i] = pers[rec_index];
        pers[rec_index] = temp;
    }
    return;
}

int main() {
```

```
    data person[NUM] = {"Michael", 18, "David", 16, "Johnson", 22, "Ben",
18,
                        "Steve",   25, "Amy",   15, "Hart",    17, "Zxh",
13,
                        "Cathy",   9,  "Lee",   21};
    printf(
        "These person datas sorted by age in increasing sequence are as "
        "follows.\n");
    sort_age(person, NUM);
    for (int i = 0; i < 10; i++) {
        printf("Name: %-10s Age: %3d\n", person[i].name, person[i].age);
    }
    sort_name(person, NUM);
    printf(
        "These person datas sorted by name in increasing sequence are as "
        "follows.\n");
    for (int i = 0; i < 10; i++) {
        printf("Name: %-10s Age: %3d\n", person[i].name, person[i].age);
    }
    return 0;
}
```

HW12——Linked List

Modify Program 13.3 to prompt the user for a name. Have the program search the existing list for the entered name. If the name is in the list, display the corresponding phone number; otherwise display the message: The name is not in the current phone directory.

Example program code: Csource.rar

```
#include <stdio.h>
#include <string.h>
#define MAXNAME 30
#define MAXPHONE 15

struct TeleType {
    char name[MAXNAME];
    char phoneNum[MAXPHONE];
    struct TeleType* nextaddr;
};

void search(struct TeleType*, char*); /* function prototype */
int len_cmp(int, int);

int main() {
    printf(
        "If you want to search a phone number from someone in the current
"
        "phone directory\n");
    printf("Please type in a name then press ENTER to continue:\n");
    char input[MAXNAME];
```

```
    memset(input, 0, sizeof(input));
    for (int i = 0; i < MAXNAME; i++) {
        scanf("%c", &input[i]);
        if (input[i] == '\n') {
            input[i] = '\0';
            break;
        }
    }

    struct TeleType t1 = {"Acme, Sam", "(555) 898 2392"};
    struct TeleType t2 = {"Dolan, Edith", "(555) 682 3104"};
    struct TeleType t3 = {"Lanfrank, John", "(555) 718 4581"};
    struct TeleType* first; /* create a pointer to a structure */

    first = &t1;        /* store t1's address in first */
    t1.nextaddr = &t2;  /* store t2's address in t1.nextaddr */
    t2.nextaddr = &t3;  /* store t3's address in t2.nextaddr */
    t3.nextaddr = NULL; /* store the NULL address in t3.nextaddr */

    search(first, input); /* send the address of the first structure */
    return 0;
}

void search(struct TeleType* contents,
            char* search_name) /* contents is a pointer */
{                              /* to a structure of type TeleType */
    while (contents != NULL) { /* search till end of linked list */
        if (!memcmp(search_name, contents->name,
                    sizeof(char) *
                        len_cmp(strlen(search_name), strlen(contents-
>name)))) {
            printf("His/Her phone number is %s.\n", contents->phoneNum);
            return;
        }
        contents = contents->nextaddr; /* get next address */
    }
    printf("The name is not in the current phone directory.\n");
    return;
}

int len_cmp(int len1, int len2) { return len1 > len2 ? len1 : len2; }
```

HW13——Command-line Arguments

Write a calculator program that accepts an arithmetic expression as command-line arguments, e.g. 2 + 3. The program should perform the corresponding calculation, and display the result: 2 + 3 = 5. There are 5 types of operations: 2 + 3 for addition; 4 - 5 for subtraction; 4 x 5 for multiplication; 4 / 2 for division; 4 % 3 for modulus.

```
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <string.h>

long long num[2], cnt = 0;

void ReadNumber(const char* str) {
    long long judge = 1; /*To judge whether the number is positive or
negative*/
    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == '-') {
            judge = -1;
        } else {
            /*Using bit operator to convert, store and update the number
from
             * argv*/
            num[cnt] = (num[cnt] << 3) + (num[cnt] << 1) + (str[i] ^ '0');
        }
    }
    /*Switch to store number in another long long variable automatically*/
    num[cnt++] *= judge;
    return;
}

void Calculation(char opt) {
    switch (opt) {
        case '+':
            printf("%lld + %lld = %lld\n", num[0], num[1], num[0] +
num[1]);
            break;
        case '-':
            printf("%lld - %lld = %lld\n", num[0], num[1], num[0] -
num[1]);
            break;
        case 'x':
            printf("%lld x %lld = %lld\n", num[0], num[1], num[0] *
num[1]);
            break;
        case '/':
            if (num[1] == 0) {
                printf("Number cannot be divided by zero.\n");
                exit(2);
            } else {
                printf("%lld / %lld = %lld\n", num[0], num[1], num[0] /
num[1]);
            }
            break;
        case '%':
            if (num[1] == 0) {
                printf("Number cannot be divided by zero.\n");
                exit(2);
            } else {
                printf("%lld %% %lld = %lld\n", num[0], num[1],
                       num[0] % num[1]);
            }
            break;
```

```
        default:
            printf(
                "You might enter a wrong operator, please check your
input.\n");
            exit(3);
    }
    return;
}

/*The program receives 4 arguments when it runs successfully
  These 4 arguments are:
  1. The address to this program
  2. The first number
  3. The operator
  4. The second number
*/
int main(int argc, char* argv[]) {
    if (argc < 4) {
        printf("The number of arguments you entered is not enough.\n");
        exit(1);
    } else {
        ReadNumber(*(argv + 1));
        ReadNumber(*(argv + 3));
        printf("The corresponding calculation is as follow:\n");
        Calculation(*(argv + 2)[0]);
    }
    return 0;
}
```

HW14——Operator Overloading in C++

Write a C++ program to overload the + operator, to add two Time objects.

1. Define the Time class which should have three attributes of int data type: hour, minute, second, with 0 as their default value.
2. Define a member function showTime() to show the Time object as 11:45:33
3. Overload the + operator, to get the summation of two Time objects with the correct forms: hour(0-23), minute(0-59), second(0-59). For example, t1 is 1:50:30 , t2 is 2:15:25, then t1 + t2 will be 4:5:55

```
#include <iostream>

using namespace std;

class Time {
   private:
    int hour = 0;
    int minute = 0;
    int second = 0;

   public:
    void showTime(void) {
```

```
        cout << "The time is:" << endl;
        cout << hour << ":";
        cout << minute << ":";
        cout << second << endl;
        return;
    }

    void getTime(void) {
        cout << "Please enter a time data(format:hour minute second)" <<
endl;
        cin >> hour >> minute >> second;
        return;
    }

    Time operator+(Time const &another) {
        Time result;
        result.second = (this->second + another.second) % 60;
        result.minute = (this->second + another.second) / 60;
        result.minute += (this->minute + another.minute) % 60;
        result.hour = (this->minute + another.minute) / 60;
        result.hour += (this->hour + another.hour);
        return result;
    }
};

int main(void) {
    Time t1, t2;
    t1.getTime();
    t2.getTime();
    (t1 + t2).showTime();
    return 0;
}
```