ルビス　ルスファン　アンシャー
15_15953

課題１（構造体の配列の並べ替え）

ソースコード名：sort.c

目的：関数ポインターを用いて、配列を並べ替える。

プログラムは 5 個の関数から成り立っている。

最初に hdata という構造体を宣言する。

```
typedef struct{
        int id;
        float height;
} hdata;
```

swap 関数は２つの hdata の順番を交換する

```
void swap(hdata *x, hdata *y){
        hdata temp;

        temp = *x;
        *x = *y;
        *y = temp;
}
```

compar_id は 2 個の hdata の id を比較する関数

```
int compar_id(const hdata *a, const hdata *b){
        if ((a -> id) < (b -> id)){
                return 1;
        } else return -1;
}
```

compar_height は 2 個の hdata の height を比較する

```
int compar_height(const hdata *a, const hdata *b){
        if ((a -> height) < (b -> height)){
```

```
                return 1;
        } else return -1;
}
```

sort 関数は hdata の配列、配列の長さと関数ポインタを受け取り、関数ポインターによって、配列をソートする。compar_id を受け取れば、id を昇順にソートし、compar_height を受け取る場合、height を昇順にソートする。

```
int sort(hdata *base, int num, int (*compar)(const hdata *a, const hdata *b)){
        int i, j;
        //ソートはバブルソートである
        for (i = 0; i < num; i++){
                int k = num - 1;
                for (j = 0; j < k; j++){
                        //base[j+1]は base[j]より小さいとき、スワップする
                        if (compar(&base[j+1], &base[j]) == 1){
                                swap(&base[j+1], &base[j]);
                        }
                }
                k = k - 1;
        }
        return 0;
}
```

main 関数で配列を定義して、ソートを実行する。その後、実行結果を表示する。

```
int main(){
        hdata base[] = {
                {4, 154},
                {2, 166},
                {8, 124},
                {1, 176}
        };

        sort(base, 4, compar_id);
        printf("compare id¥n");
```

```
	for (int i = 0; i < 4; i++){
		printf("%d:%d(id=%f)¥n",i+1, base[i].id, base[i].height);
	}

	printf("¥n");

	sort(base, 4, compar_height);
	printf("compare height¥n");
	for (int i = 0; i < 4; i++){
		printf("%d:%d(id=%f)¥n",i+1, base[i].id, base[i].height);
	}

	return 0;
}
```

課題２（blt の実装）

ソースコード名：blt.s

```
        .data
A:
        .asciiz "true"
B:
        .asciiz "false"

        .text
main:
        li $t1, 4
        li $t2, 6
        slt $t0, $t1, $t2          # $t1 < $t2 をチェックする
        bne $t0, $zero, label      # $t0 != 0 であれば、label にジャンプする
        li $v0, 4
        la $a0, B
        syscall                                  # false を print
        j end

label:
        li $v0, 4
        la $a0, A
        syscall                                  # true を print

end:
        jr $ra
```

課題 3（２つの配列の要素の和）

ソースコード名：addition.s

```
        .data
A:
        .word 1 2 3 4
B:
        .word 5 6 7 8

        .text
main:
        li $t0, 0                   #A のインデックスは先頭(0)から始まる
        li $t1, 12                  #B のインデックスは末尾(12)から始まる
        # word の長さは 4 なので、足し算の回数を数えるために 4 を$t6 に入
        れる。
        li $t6, 4
loop:
        beq $t6, $zero, end         # $t6 が 0 になれば、足し算が終わる
        lw $t3, A($t0)
        lw $t4, B($t1)
        add $t5, $t3, $t4

        #足し算の結果を表示する
        li $v0, 1
        move $a0, $t5
        syscall

        #A のインデックスを 1 個前に進み、B のインデックスが 1 個後ろ動く
        addi $t0, $t0, 4
        addi $t1, $t1, -4
        addi $t6, $t6, -1
        j loop
end:
        jr $ra
```

課題 4（最大値の検索）

ソースコード名：max.s

```
        .data
A:
        .word 2 7 113 1 3 9

        .text
main:
        li $t0, 0               # 最大初期値を 0 にする
        li $t1, 0               # インデックスのためのレジスター

loop:  (最大値を求めるために、word をループでチェックする)
        lw $t2, A($t1)          # 配列からの値を$t2 に入れる
        beq $t2, $zero, end     # 値が 0 ならば、配列のチェックが完了

        # $t0 と$t2 を比較し、$t2 が大きい場合、スワップする
        blt $t0, $t2, change
        addi $t1, $t1, 4
        j loop

change:                         # $t0 に新しい最大値を代入
        move $t0, $t2
        addi $t1, $t1, 4        # インデックスを 4 で足す
        j loop

end:
        # 最大値を表示する
        li $v0, 1
        move $a0, $t0
        syscall
        j $ra
```

課題 5（ble の実装）

ソースコード名：ble.s

```
        .data
A:
        .asciiz "true"
B:
        .asciiz "false"

        .text
main:
        li $t0, 4
        li $t1, 3
        sub $t2, $t0, $t1                    # $t2 = $t0 - $t1

        # $t2 < 1 をチェックした後、$t3 にその結果($t2 < 1 の場合、$t3 = 1, $t2 !< 1 の場合、$t3 = 0)を代入する。$t3 は 0 の場合、false を表示して、そうでない場合、true を表示。
        slti $t3, $t2, 1
        bne $t3, $zero, label

        li $v0, 4
        la $a0, B
        syscall                                    # print false
        j end
label:
        li $v0, 4
        la $a0, A
        syscall                                    # print true
end:
        jr $ra
```