ルビス　ルスファン　アンシャー
15_15953

課題 1

ファイル名：montecarlo.c
目的　　：モンテカルロ法による円周率の計算

プログラムを int pi(int n, double *pi)関数と main 関数に分けられる。

関数 pi で円周率を推定し、推定値が *pi に代入される。pi を推定するために、n 回乱数(x, y)をとり、$x^2 + y^2 < 100000000$ の条件を満たすものを数え、その数を m とする。$pi = 4 \times m/n$で求める。

```
int pi(int n, double *pi){
        int i;                                  //for 文のため
        int m = 0;

        for (i = 0; i < n; i++){
                //乱数を 1-1000 の間からとる
                int x = rand() % 10000 + 1;
                int y = rand() % 10000 + 1;

                if ((x*x)+(y*y) < 100000000){
                //n 個の乱数(x,y)のなか、以上の条件を満たすものを数える。
                        m = m+1;
                }
        }
        if (m > 0){
                *pi = 4*((double)m)/n;          //求めた pi の値を*pi に代入する
                return 0;
        } else {
                return -1;
        }
}
```

main 関数で n を決めて（大きいほど、真の pi 値に近い）、初期値 0 の double *b を宣言する。

```
int main(){
        int n = 100000;
        double *b;
        time_t t;

        //プログラムを run するごとに、違う結果が出るように、以下の文が必要
        srand((unsigned) time(&t));

        for (int i = 0; i < 5; i++){
                *b = 0;
                printf("initial: %f¥n", *b);          //*b の初期値を表示する
                pi(n, b);                             //pi を実行
                printf("result: %f¥n", *b);           //pi 実行後の*b を表示
        }
        return 0;
}
```

例：

```
initial: 0.000000
result: 3.139800
initial: 0.000000
result: 3.133120
initial: 0.000000
result: 3.141320
initial: 0.000000
result: 3.143000
initial: 0.000000
result: 3.137440
```

課題 2

ファイル名：sort.c

目的　　：int の配列をソートする

プログラムが void sort_int_array(int *ary, int ary_size)関数と main 関数からなる。

関数 sort_int_array はバブルソートで配列をソートする。

```
void sort_int_array(int *ary, int ary_size){
        int i, j;
        int temp;

        for (i = 0; i < ary_size; i++){
                int k = ary_size - 1;           //配列の末尾を比較する必要がないので、ary_size - 1 にする
                for (j = 0; j < k; j++){
                        if (ary[j+1] < ary[j]){
                                //ary[j]と ary[j+1]をスワップする
                                temp = *(ary+j);
                                *(ary+j) = *(ary+j+1);
                                *(ary+j+1) = temp;
                        }
                }
                k = k - 1;
        }
}
```

main 関数でランダムな整数の配列を生じ、ソート関数を実行して、ソート後の配列を表示する。

```
int main(){
        int size = 10;
        int rand_array[size];
        time_t t;
```

```
	srand((unsigned) time(&t));

	//5 回ランダムな配列を生成しソートする
	for (int k = 0; k < 5; k++){
		for (int j = 0; j < size; j++){
		//ランダムな整数の配列を作る
			rand_array[j] = rand() % 100 + 1;
		}

		sort_int_array(rand_array, size);		//ソートを実行

		//ソート後を表示する
		printf("sorted array: ");
		for (int i = 0; i < size; i++){
			printf("%d ", *(rand_array+i));
		}
		printf("¥n");
	}
	return 0;
}
```

例：

initial array: 32 70 81 19 99 34 8 68 94 49

sorted array: 8 19 32 34 49 68 70 81 94 99

initial array: 64 100 80 62 88 86 55 22 98 97

sorted array: 22 55 62 64 80 86 88 97 98 100

initial array: 60 32 93 76 53 46 88 67 43 67

sorted array: 32 43 46 53 60 67 67 76 88 93

initial array: 45 95 36 44 1 27 12 33 10 73

sorted array: 1 10 12 27 33 36 44 45 73 95

課題 3

ファイル名：concat.c

目的：２つの文字列を連結

プログラムは mystrlen 関数、concat 関数と main 関数からなる。

mystrlen はある文字列の長さを出力する関数である。

```
int mystrlen(char *src){
        int i = 0;

        //格文字列の一番後ろの様子は常に'¥0'があるので、それを見つけたら、文字
        列が終わる
        while(src[i] != '¥0'){
                i++;
        }
        return i;
}
```

concat 関数は２つの文字列を連結する。

```
void concat(char *target, char *src1, char *src2){
        int i = 0;
        int s;
        int ln1 = mystrlen(src1);
        int ln2 = mystrlen(src2);

        if (src1[0] == '¥0' && src2[0] == '¥0'){
        //両方の文字列は空の場合、連結結果も空になる、それを*target に代入
            *target = '¥0';
        } else if (src1[0] == '¥0'){
        //src1 は空の場合、src2 の各文字を*target に代入
                for (s = 0; src2[s] != '¥0'; s++){
                    *(target+s) = *(src2+s);
                }
```

```
        } else if (src2[0] == '¥0'){
        //src2 は空の場合、src1 の各文字を*target に代入
                for (s = 0; src1[s] != '¥0'; s++){
                    *(target+s) = *(src1+s);
                }
        } else if (ln1 >= ln2){
                while (src2[i] != '¥0'){
                //src1 は src2 より長い場合、*target に src1 と src2 の文字を同時
                に src2 が終わるまで入れる
                        target[i] = src1[i];
                        target[ln1 + i] = src2[i];
                        i++;
                }
                while (src1[i] != '¥0'){
                //残った src1 の文字列を加える
                        target[i] = src1[i];
                        i++;
                }
        } else {
        //src2 は src1 より長い場合
                while (src1[i] != '¥0'){
                        target[i] = src1[i];
                        target[ln1 + i] = src2[i];
                        i++;
                }
                while (src2[i] != '¥0'){
                        target[ln1 + i] = src2[i];
                        i++;
                }
        }
}
```

main 関数が初期の文字列と連結したい文字列を宣言し、concat を実行して、最後に concat 後の文字列を表示する。

```
int main(){
	//連結の結果のための空文字列
	char s1[100];
	char s2[100];
	char s3[100];
	char s4[100];

	char *emp = "¥0";
	char *str1 = "awiefbaow";
	char *str2 = "oaiwenpoarw";

	concat(s1, emp, emp);
	//両方の文字列は空の場合
	printf("str1: %s, str2: %s, concat: %s¥n", emp, emp, s1);

	concat(s2, emp, str1);
	//一方の文字列は空
	printf("str1: %s, str2: %s, concat: %s¥n", emp, str1, s2);

	concat(s3, str2, emp);
	printf("str1: %s, str2: %s, concat: %s¥n", str2, emp, s3);

	concat(s4, str1, str2);
	printf("str1: %s, str2: %s, concat: %s¥n", str1, str2, s4);

	return 0;
}
```