プログラミング基礎

2023年度1Q　火曜日３,４時限(10:45~12:25)
金曜日1~４時限(　8:50~12:25)

工学院　情報通信系

中山実，渡辺義浩
伊藤泉，小杉哲
TA:小泊大輝，千脇彰悟

5/16(火) 10:45~12:25

- 第10回「探索」

1. 線形探索
2. 二分探索
3. 乱数の発生

線形探索（逐次検索）

- 学籍番号4391の学生の点数は？
- キー：学籍番号4391
- 表の先頭から末尾に向けてキーと一致する項目を検索

学籍番号	5267	4295	2283	4329	4391	3312	4311	2349	4432	..	5334
点数	55	75	90	66	75	94	85	89	78		79

データ数：n

- 最良：1回目で発見
- 最悪：n回目で発見
- 平均：n/2回目で発見

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4320, 5560, 4340, 4360, 5603, 6645, 4090, 6403,
                    5440, 4375, 5466, 6346, 3330, 6703, 7645, 4790, 4403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int i, key;

    printf("ID?");
    scanf("%d", &key);
    for (i = 0; i < SIZE; i++)
    {
        if (id[i] == key)
        {
            printf("key = %04d, score = %3d¥n", key, score[i]);
            return 0;
        }
    }
    printf("key = %04d is not found¥n", key);
    return 0;
}
```

- データを配列の初期値として与える
- id: 学籍番号，score: 得点

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4320, 5560, 4340, 4360, 5603, 6645, 4090, 6403,
                    5440, 4375, 5466, 6346, 3330, 6703, 7645, 4790, 4403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int i, key;

    printf("ID?");
    scanf("%d", &key);
    for (i = 0; i < SIZE; i++)
    {
        if (id[i] == key)
        {
            printf("key = %04d, score = %3d¥n", key, score[i]);
            return 0;
        }
    }
    printf("key = %04d is not found¥n", key);
    return 0;
}
```

- 検索する学籍番号の入力

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4320, 5560, 4340, 4360, 5603, 6645, 4090, 6403,
                    5440, 4375, 5466, 6346, 3330, 6703, 7645, 4790, 4403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int i, key;

    printf("ID?");
    scanf("%d", &key);
    for (i = 0; i < SIZE; i++)
    {
        if (id[i] == key)
        {
            printf("key = %04d, score = %3d¥n", key, score[i]);
            return 0;
        }
    }
    printf("key = %04d is not found¥n", key);
    return 0;
}
```

- keyと一致する項目を逐次検索

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4320, 5560, 4340, 4360, 5603, 6645, 4090, 6403,
                    5440, 4375, 5466, 6346, 3330, 6703, 7645, 4790, 4403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int i, key;

    printf("ID?");
    scanf("%d", &key);
    for (i = 0; i < SIZE; i++)
    {
        if (id[i] == key)
        {
            printf("key = %04d, score = %3d¥n", key, score[i]);
            return 0;
        }
    }
    printf("key = %04d is not found¥n", key);
    return 0;
}
```

- 最後まで探してなければ，ないと答える

5/16(火) 10:45~12:25

- 第10回「探索」

1. 線形探索
2. 二分探索
3. 乱数の発生

二分探索

- 要素数がnの配列a
- 要素は昇順に並んでいる
- 配列からkeyを探索する

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

二分探索

- 探索範囲の先頭: lw (探索開始時、0)
- 探索範囲の末尾: up (探索開始時、n-1)
- 探索範囲の中央: md (探索開始時、(n-1)/2(の整数部))

	lw				md					up
index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

二分探索

- keyとa[md]が一致しなかった場合，以下のように探索範囲を縮小する
- key > a[md]のとき
  - lw = md + 1, up = up → md = (lw + up) / 2 (の整数部)
- key < a[md]のとき
  - lw = lw, up = md – 1 → md = (lw + up) / 2 (の整数部)
- 以下のいずれかが成立した場合に終了
  - a[md]とkeyが一致
  - 探索範囲がなくなる (lw > up)

lw　　md　　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

二分探索

- key = 13を探索する場合

lw　　md　　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

lw　　md　　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

lw　md　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

探索3回で“13”を発見

二分探索

- key = 2を探索する場合

lw　　md　　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

lw　　md　　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

lw md　up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

二分探索

- key = 2を探索する場合

lw md up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

lw md up

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

up lw

index	0	1	2	3	4	5	6	7	8	9	10
value	1	3	5	7	9	11	13	15	17	19	21

探索範囲がなくなり、”2”と一致するものが見つからず終了 (探索回数4回と数える)

- データを配列の初期値として与える
- id: 学籍番号(ソートされている)
- score: 得点

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4520, 4560, 4840, 5360, 5603, 5945, 6090, 6403,
                    6440, 6975, 7066, 7146, 7330, 7703, 7945, 8090, 8403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int key, lw, md, up;

    printf("ID?");
    scanf("%d", &key);
    lw = 0, up = SIZE - 1;
    while (lw <= up)
    {
        md = (lw + up) / 2;
        if (id[md] == key)
        {
            printf("key = %04d ¥nscore = %3d¥n", key, score[md]);
            return 0;
        }
        else if (id[md] < key)
            lw = md + 1;
        else
            up = md - 1;
    }
    printf("key =%04d is not found¥n", key);
    return 0;
}
```

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4520, 4560, 4840, 5360, 5603, 5945, 6090, 6403,
                    6440, 6975, 7066, 7146, 7330, 7703, 7945, 8090, 8403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int key, lw, md, up;

    printf("ID?");
    scanf("%d", &key);
    lw = 0, up = SIZE - 1;
    while (lw <= up)
    {
        md = (lw + up) / 2;
        if (id[md] == key)
        {
            printf("key = %04d ¥nscore = %3d¥n", key, score[md]);
            return 0;
        }
        else if (id[md] < key)
            lw = md + 1;
        else
            up = md - 1;
    }
    printf("key =%04d is not found¥n", key);
    return 0;
}
```

- 検索する学籍番号の入力

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4520, 4560, 4840, 5360, 5603, 5945, 6090, 6403,
                    6440, 6975, 7066, 7146, 7330, 7703, 7945, 8090, 8403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int key, lw, md, up;

    printf("ID?");
    scanf("%d", &key);
    lw = 0, up = SIZE - 1;
    while (lw <= up)
    {
        md = (lw + up) / 2;
        if (id[md] == key)
        {
            printf("key = %04d ¥nscore = %3d¥n", key, score[md]);
            return 0;
        }
        else if (id[md] < key)
            lw = md + 1;
        else
            up = md - 1;
    }
    printf("key =%04d is not found¥n", key);
    return 0;
}
```

- 探索範囲の上限と下限の中間値
- 小数点以下切り捨て

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4520, 4560, 4840, 5360, 5603, 5945, 6090, 6403,
                    6440, 6975, 7066, 7146, 7330, 7703, 7945, 8090, 8403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int key, lw, md, up;

    printf("ID?");
    scanf("%d", &key);
    lw = 0, up = SIZE - 1;
    while (lw <= up)
    {
        md = (lw + up) / 2;
        if (id[md] == key)
        {
            printf("key = %04d ¥nscore = %3d¥n", key, score[md]);
            return 0;
        }
        else if (id[md] < key)
            lw = md + 1;
        else
            up = md - 1;
    }
    printf("key =%04d is not found¥n", key);
    return 0;
}
```

- 中間値の学籍番号と検索するkeyが一致すれば終了

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4520, 4560, 4840, 5360, 5603, 5945, 6090, 6403,
                    6440, 6975, 7066, 7146, 7330, 7703, 7945, 8090, 8403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int key, lw, md, up;

    printf("ID?");
    scanf("%d", &key);
    lw = 0, up = SIZE - 1;
    while (lw <= up)
    {
        md = (lw + up) / 2;
        if (id[md] == key)
        {
            printf("key = %04d ¥nscore = %3d¥n", key, score[md]);
            return 0;
        }
        else if (id[md] < key)
            lw = md + 1;
        else
            up = md - 1;
    }
    printf("key =%04d is not found¥n", key);
    return 0;
}
```

- 中間値の学籍番号が，検索するkeyより小さければ下限を引き上げ，大きければ上限を引き下げる

```c
#include <stdio.h>
#define SIZE 18

int main(void)
{
    int id[SIZE] = {4430, 4520, 4560, 4840, 5360, 5603, 5945, 6090, 6403,
                    6440, 6975, 7066, 7146, 7330, 7703, 7945, 8090, 8403};
    int score[SIZE] = {90, 68, 57, 90, 43, 63, 89, 49, 77,
                       55, 68, 98, 76, 86, 83, 79, 78, 34};
    int key, lw, md, up;

    printf("ID?");
    scanf("%d", &key);
    lw = 0, up = SIZE - 1;
    while (lw <= up)
    {
        md = (lw + up) / 2;
        if (id[md] == key)
        {
            printf("key = %04d ¥nscore = %3d¥n", key, score[md]);
            return 0;
        }
        else if (id[md] < key)
            lw = md + 1;
        else
            up = md - 1;
    }
    printf("key =%04d is not found¥n", key);
    return 0;
}
```

実行結果例

```
ID?7330
key = 7330
score = 86
```

5/16(火) 10:45~12:25

- 第10回「探索」

1. 線形探索
2. 二分探索
3. 乱数の発生 (参考)

乱数の発生（参考）

- C言語では、乱数を発生させる関数が用意されている
- データの生成や解析、実験の制御などでよく用いられる

- int rand(void)
  - 0～RND_MAX(4byte, 2147483647までの数値を返す
  - rand()%10 +1 で1~10の数値を返す
  - (double)rand()/RNAD_MAXで０以上１以下の数値を返す
  - ただし, rand()は毎回同じ乱数系列を発生する

- void srand(unsigned seed)
  - rand関数で発生させる乱数系列を変更する

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

int main(void)
{
    int i, rand1, rand2;
    double rand3;

    for (i = 0; i < 5; i++)
    {
        rand1 = rand();
        rand2 = rand1 % 10 + 1;
        rand3 = (double)rand1 / RAND_MAX;
        printf("rand =%11d, %2d, %3.6lf¥n", rand1, rand2, rand3);
    }
    return 0;
}
```

- 標準ライブラリstdlib.hをインクルード
- 乱数発生

実行結果例（1回目）

```
rand =         41,  2, 0.001251
rand =      18467,  8, 0.563585
rand =       6334,  5, 0.193304
rand =      26500,  1, 0.808741
rand =      19169, 10, 0.585009
```

実行結果例（2回目）

```
rand =         41,  2, 0.001251
rand =      18467,  8, 0.563585
rand =       6334,  5, 0.193304
rand =      26500,  1, 0.808741
rand =      19169, 10, 0.585009
```

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

int main(void)
{
    int i;

    printf("rand1 = ");
    srand(1);
    for (i = 0; i < 5; i++)
        printf(" %2d", rand() % 10 + 1);
    printf("¥nrand1 = ");
    srand(1);
    for (i = 0; i < 5; i++)
        printf(" %2d", rand() % 10 + 1);

    printf("¥nrand2 = ");
    srand(2);
    for (i = 0; i < 5; i++)
        printf(" %2d", rand() % 10 + 1);

    return 0;
}
```

- 乱数系列の設定
- シード値が同じ場合、乱数系列も同じ
- シード値を変更

実行結果

```
rand1 =   2  8  5  1 10
rand1 =   2  8  5  1 10
rand2 =   6  7  9  6  5
```

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

int main(void)
{
    int i;

    printf("rand1 = ");
    srand((unsigned)time(NULL));
    for (i = 0; i < 5; i++)
        printf(" %2d", rand() % 10 + 1);

    sleep(1);
    printf("¥nrand2 = ");
    srand((unsigned)time(NULL));
    for (i = 0; i < 5; i++)
        printf(" %2d", rand() % 10 + 1);

    return 0;
}
```

- time_t time(NULL)
  - 時刻情報を取得
  - 1970年1月1日 00:00:00からの経過時間(秒)を取得
- unsigned int sleep(unsigned int seconds)
  - 指定した秒の間、処理を中断
- 時刻情報でシード値を設定
- 1秒ごとに異なる乱数系列を設定できる

実行結果例

```
rand1 =  10  2  6  9  3
rand2 =   3  2  1  4  8
```