6/18(水) 情報Ⅰ　～総復習その①～　　クラスルーム→　　モグラノート→　　←モバイル Wi-Fi→

◎連絡事項◎

① スライドを1つ作る課題が出ています！詳細は Google クラスルームにて。提出期限は、6/23(月)23 時 59 分まで。

② 期末テスト範囲　→　プログラミング分野のみ

・集計　・順位付け　・最大値、最小値　・探索(線形探索、二分探索)　・整列(隣接交換、選択、挿入)　・総合問題

☆集計

右の文は、goukei に〈変数〉を次々に加算し、合計を求めるプログラムである。

```
〈変数〉を○から□まで 1 ずつ増やしながら繰り返す：
  goukei = goukei +〈変数〉
```

1　次のプログラムは、1 から 5 までの自然数の和を表示するプログラムである。

```
（1）goukei = 0
（2）i を 1 から［ア］まで 1 ずつ増やしながら繰り返す：
（3）  goukei =［イ］
（4）  表示する（"合計は", goukei）
```

【出力結果】 合計は 15

［ア］に適するものは 1

① 4　② 5　③ 6　④ 7

［イ］に適するものは 6

⑤ goukei + 1　⑥ goukei + i

⑦ i　⑧ goukei * i

2　次のプログラムは、配列 Data 内の値を集計するプログラムである。

```
（1）Data = [3, 5, 4, 1, 7]
（2）n = 要素数(Data)
（3）sum = 0
（4）i を 0 から［ア］まで 1 ずつ増やしながら繰り返す：
（5）  sum =［イ］
（6）  表示する（"合計は", sum）
```

【出力結果】 合計は 20

［ア］に適するものは 1

① n − 1　② n

③ n + 1　④ n + 2

［イ］に適するものは 6

⑤ sum + 1　⑥ sum + Data[i]

⑦ sum + i　⑧ Data[i]

☆順位付け

右の文は、整列された配列 Data の値にそれぞれ順位付けするプログラムである。

```
i を 0 から□まで 1 ずつ増やしながら繰り返す：
  表示する（i +1, "位：", Data[i]）
```

3　次のプログラムは、降順に整列された配列 Data 内で大きいものから順位を付けるプログラムである。

```
（1）Data = [100, 70, 50, 10]
（2）i を 0 から［ア］まで 1 ずつ増やしながら繰り返す：
（3）  表示する（［イ］, "位：", Data[i], "点"）
```

【出力結果】 1 位：100 点
2 位：70 点
3 位：50 点
4 位：10 点

［ア］に適するものは 2

① 2　② 3　③ 4　④ 5

［イ］に適するものは 7

⑤ i − 1　⑥ i

⑦ i + 1　⑧ i + 2

☆交換

右の文は、2 つの変数 a,b を交換（こうかん）するプログラムである。

このプログラム内で使われる変数 temp を一時変数（いちじへんすう）（テンポラリ変数）という。

```
temp = a
a = b
b = temp
```

4　次のプログラムは、配列 Data 内の 5 と 3 を交換するプログラムである。

```
（1）Data = [5, 3]
（2）temp = Data [0]
（3）Data[0] =［ア］
（4）Data[1] =［イ］
（5）表示する（Data）
```

【出力結果】 3, 5

［ア］に適するものは 2

① Data[0]　② Data [1]　③ temp　④ 5

［イ］に適するものは 7

⑤ Data [0]　⑥ Data [1]　⑦ temp　⑧ 3

☆最大値・最小値

右の文は、配列 Data 内の最大値を求めるプログラムである。
配列を左から見ていき、より大きいものがあれば、変数 max をそれに更新する。

```
i を○から□まで 1 ずつ増やしながら繰り返す：
｜　もし max < Data[i]ならば：
└　└　max = Data[i]
```

5　次のプログラムは、配列 Data 内の最大値を求めるプログラムである。

```
（1）Data = [1, 6, 4, 8, 10]
（2）n = 要素数(Data)
（3）max = Data[0]
（4）i を 1 から［ア］まで 1 ずつ増やしながら繰り返す：
（5）｜　もし［イ］ならば：
（6）└　└　max = Data[i]
（7）表示する（”最大値は”, max）
```

【出力結果】　最大値は 10

［ア］に適するものは 1

① n − 1　　② n
③ n + 1　　④ n + 2

［イ］に適するものは 5

⑤ max < Data[i]
⑥ max > Data[i]

6　次のプログラムは、配列 Data 内の最小値を求めるプログラムである。

```
（1）Data = [10, 6, 4, 2, 1, 5]
（2）n = 要素数(Data)
（3）min = Data[0]
（4）i を 1 から［ア］まで 1 ずつ増やしながら繰り返す：
（5）｜　もし［イ］ならば：
（6）└　└　min =［ウ］
（7）表示する（”最小値は”, min）
```

【出力結果】　最小値は 1

［ア］に適するものは 1

① n − 1　　② n
③ n + 1　　④ n + 2

［イ］に適するものは 6

⑤ min < Data[i]
⑥ min > Data[i]

［ウ］に適するものは 8

⑦ Data[i − 1]
⑧ Data[i]
⑨ Data[i + 1]

☆線形探索法

① 複数のデータの中から目的のデータを探し出すことを、**探索**という。

② 配列を先頭から順にみていきながら、探索値に一致するデータを探し出す探索方法を、**線形探索法**という。

② 右の文は、線形探索法のプログラムである。
記号「＝＝」は、「一致する」という意味。
探索値 x が見つかったら、繰り返しを終了する。

```
i を○から□まで 1 ずつ増やしながら繰り返す：
｜　もし Data[i] == x ならば：
｜　｜　表示する(〜〜, “番目”)
└　└　繰り返しを抜ける
```

実験 7　次の配列において、(1)〜(4)は左から何番目かを線形探索法で調べよう。(モグラノートで実験)

10,　16,　45,　57,　68,　75,　91

(1)　75　　(2)　45　　(3)　91　　(4)　57

8　次のプログラムは、配列 Data から線形探索法で変数 x に入力された値を探すプログラムである。

```
（1）Data = [10, 16, 45, 57, 68, 75, 91]
（2）n = 要素数(Data)
（3）x = 【外部からの入力】
（4）i を 0 から n − 1 まで 1 ずつ増やしながら繰り返す：
（5）｜　もし［ア］ならば：
（6）｜　｜　表示する（x, “は”,［イ］, ”番目に存在”）
（7）└　└　繰り返しを抜ける
```

【出力結果】　75 を入力　→　75 は 6 番目に存在
45 を入力　→　45 は 3 番目に存在
91 を入力　→　91 は 7 番目に存在
57 を入力　→　57 は 4 番目に存在

［ア］に適するものは 1

① Data[i] == x
② Data[i] != x
③ Data[i] < x
④ Data[i] > x

［イ］に適するものは 7

⑤ i − 1
⑥ i
⑦ i + 1
⑧ i + 2

6/18(水) 情報Ⅰ　～総復習その②～

☆二分探索法

① 探索範囲を半分に絞り込むことを繰り返して、
目的のデータを探す方法を、**二分探索法**(にぶたんさくほう)という。
なお、事前にデータを昇順または降順に並べておく。

② 右の文は、二分探索法のプログラムである。
変数 hidari から migi までが、探索範囲を表す。
変数 aida は、探索範囲の中央を表す。
この中央値が探索値 x と一致するかを調べる。
一致しないなら、探索範囲を右半分 or 左半分にする。

```
hidari <= migi の間繰り返す：
｜　aida = 〈中央値〉
｜　｜　もし Data[aida] == x ならば：
｜　｜　表示する(～～, "番目")
｜　そうでなくもし ～～ ならば：
｜　｜　migi = aida – 1
｜　そうでなければ：
└　└　hidari = aida + 1
```

実験 9　次の配列において、(1)～(4)は左から何番目かを二分探索法で調べよう。(モグラノートで実験)

10,　16,　45,　57,　68,　75,　91

(1)　75　　(2)　45　　(3)　91　　(4)　57

10　次のプログラムは、配列 Data から二分探索法で変数 x に入力された値を探すプログラムである。

```
（1）Data = [10, 16, 45, 57, 68, 75, 91]
（2）hidari = 0
（3）migi = 要素数(Data) – 1
（4）x = 【外部からの入力】
（5）hidari <= migi の間繰り返す：
（6）｜　aida = [ア]
（7）｜　もし [イ] == x ならば：
（8）｜　｜　表示する（aida + 1, "番目"）
（9）｜　｜　繰り返しを抜ける
(10）｜　そうでなくもし Data[aida] < x ならば:
(11）｜　｜　[ウ]
(12）｜　そうでなければ：
(13）└　└　[エ]
```

【出力結果】　75 を入力　→　6 番目
　　45 を入力　→　3 番目

[ア]に適するものは 1
① (hidari + migi) ÷ 2
② (hidari + migi – 1) ÷ 2

[イ]に適するものは 4
③ Data[hidari]
④ Data[aida]
⑤ Data[migi]

[ウ]に適するものは 6
⑥ hidari = aida + 1
⑦ migi = aida – 1

[エ]に適するものは 9
⑧ hidari = aida + 1
⑨ migi = aida – 1

☆バブルソート（隣接交換法）

① データをある規則にしたがって整列することを**ソート**という。
小さい値から大きな値へと整列する**昇順**(しょうじゅん)と、大きな値から小さな値へと整列する**降順**(こうじゅん)がある。

② 隣同士を比較して、昇順または降順になるように交換を
繰り返す方法を、**バブルソート（隣接交換法）**(りんせつこうかんほう)という。

③ 右の文は、バブルソートのプログラムである。
変数 i と j の二重ループを繰り返す。
Data[j] ≦ Data[j+1]
　　↑この不等号は、昇順か降順かで変わる。
temp 以降の 3 行は、Data[j]と Data[j+1]を交換する。

```
i を～～しながら繰り返す：
｜　j を～～しながら繰り返す：
｜　｜　もし Data[j] ≦ Data[j+1]ならば：
｜　｜　｜　temp = Data[j]
｜　｜　｜　Data[j] = Data[j+1]
└　└　└　Data[j+1] = temp
```

実験 11　次の配列を、バブルソートで昇順に並べ替えしてみよう。(モグラノートで実験)

(1)　12,　11,　13,　5　　(2)　77,　42,　89,　58,　97,　3,　18,　62,　33,　29

12　次のプログラムは、バブルソートにより、配列 Data を昇順に並び替えるものである。

```
（1）Data = [12, 11, 13, 5]
（2）n = 要素数(Data)
（3）i を 1 から n – 1 まで 1 ずつ増やしながら繰り返す：
（4）｜ j を 0 から [ア] まで 1 ずつ増やしながら繰り返す：
（5）｜　｜ もし Data[j] > Data[j+1] ならば：
（6）｜　｜　｜　temp = Data[j]
（7）｜　｜　｜　Data[j] = [イ]
（8）└　└　└　Data[j+1] = [ウ]
（9）表示する（Data）
```

【出力結果】　5, 11, 12, 13

[ア]に適するものは 3
① n – 1
② n + 1 + i
③ n – 1 – i

[イ]に適するものは 5
④ Data[j]
⑤ Data[j + 1]
⑥ temp

[ウ]に適するものは 9
⑦ Data[j]
⑧ Data[j + 1]
⑨ temp

☆選択ソート

最小値または最大値を選択して交換することで並べ替える方法を、**選択ソート**(せんたく)という。

実験 13　次の配列を、選択ソートで昇順に並べ替えしてみよう。(モグラノートで実験)

(1)　12,　11,　13,　5　　(2)　77,　42,　89,　58,　97,　3,　18,　62,　33,　29

14　次のプログラムは、選択ソートにより、配列 Data を昇順に並び替えるものである。

```
(1) Data = [12, 11, 13, 5]
(2) n = 要素数(Data)
(3) i を 0 から n－2 まで 1 ずつ増やしながら繰り返す:
(4) | min = i
(5) | j を i＋1 から［ア］まで 1 ずつ増やしながら繰り返す:
(6) | | もし［イ］ならば:
(7) | | | min = j
(8) | temp = Data[i]
(9) | Data[i] = ［ウ］
(10) | Data[min] = temp
(11) 表示する (Data)
```

【出力結果】　5, 11, 12, 13

［ア］に適するものは 1

① n－1　② n
③ n＋1　④ n＋2

［イ］に適するものは 5

⑤ Data[min] > Data[j]
⑥ Data[min] < Data[j]

［ウ］に適するものは 8

⑦ Data[i]
⑧ Data[min]
⑨ temp

☆挿入ソート

未整列の要素を整列済みの要素列の適切な位置に挿入することで並べ替える方法を、**挿入ソート**(そうにゅう)という。

実験 15　次の配列を、挿入ソートで昇順に並べ替えしてみよう。(モグラノートで実験)

(1)　12,　11,　13,　5　　(2)　77,　42,　89,　58,　97,　3,　18,　62,　33,　29

16　次のプログラムは、挿入ソートにより、配列 Data を昇順に並び替えるものである。

```
(1) Data = [12, 11, 13, 5]
(2) n = 要素数(Data)
(3) i を 1 から n－1 まで 1 ずつ増やしながら繰り返す:
(4) | hokan = Data[i]
(5) | j = i－1
(6) | ［ア］の間繰り返す
(7) | | Data[j+1] = Data[j]
(8) | | j = j－1
(9) | ［イ］
(10) 表示する(Data)
```

【出力結果】　5, 11, 12, 13

［ア］に適するものは 2

① j > 0　and　Data[j] > hokan
② j >= 0　and　Data[j] > hokan
③ j > 0　or　Data[j] > hokan
④ j >= 0　or　Data[j] > hokan

［イ］に適するものは 8

⑤ Data[i] = hokan
⑥ Data[i + 1] = hokan
⑦ Data[j] = hokan
⑧ Data[j + 1] = hokan

総合問題の演習　→　「実践攻略 情報Ⅰ」テキストの 94〜103 ページ