# プログラミング発展

2023年度2Q　火曜日5~7時限(13:45~16:30)
金曜日5~7時限(13:45~16:30)

工学院　情報通信系

尾形わかは，松本隆太郎，
Chu Van Thiem，　Saetia Supat
TA: 東海林郷志，千脇彰悟

**６月8日１２：００時に更新**

# １回目と２回目「プログラミング基礎復習とアルゴリズム的考え方」

0. (情報工学系計算機室利用ガイド)
1. この科目の進め方・学び方の説明
2. プログラミング基礎の復習
3. プログラミング基礎の課題プログラムとアルゴリズム的考え方の基礎

**担当教員**

尾形わかは 大岡山 連絡は東工大公式スラックで
松本隆太郎 大岡山 連絡は東工大公式スラックで
Chu Van Thiem　すずかけ台
Saetia Supat　　すずかけ台

**TA**

東海林（しょうじ）　郷志　山田研究室
千脇　彰悟　　松本研究室

# 情報工学系計算機室ルール

- ルール違反するとアカウント停止になるので、T2Scholaにあるルールを熟読すること、特に
- 計算機室内飲食禁止
- 玄関より先に傘の持ち込み禁止
- スリッパ着用
- **アカウント持っていない人は今すぐに申し出ること**

# 出席の取り扱い

- **出席はとりません。**
- レポートを締め切り時刻までに提出した場合、その回の講義に出席したと見なします。
- 病気などの事情で締め切りまでにレポート提出できない場合、その事情を証明できる書類（例えば病院、薬局の領収書、会葬礼状など）と、事情の説明をPDFにしてレポートとして提出してください。締め切りの延長を認めるために客観的証拠は必須です。

# 発熱の場合の証拠

- 学生証
- 体温計などの計測結果
- 日付（日付表示つき時計やPC画面等）

が１枚に写された写真でも可

# 課題に対する質問について

- つまずいたら、計算機室内にいるTAまたは教員に積極的に声を掛け、**講義時間中に疑問を解消するようにして下さい。**
- 原則、質問は講義時間中に対面でしてください。質問はしたいが止むを得ない事情により登校できない場合は（**学校感染症**になった等）、オンラインで受講できるように対応するので、事前に**スラック**で申し出ること。（事前または事後に、事情を説明する資料の提出を求めることがあります）

**講義の概要とねらい**

プログラミング基礎で学んだことを元に，構造体，ポインタの扱いやファイル操作などを学習する．また，実践的なプログラミングにおいて重要な計算量の概念などアルゴリズム的な考え方の基礎についても解説する．これらによって，将来研究を遂行するために必要となる実践的なプログラミング能力を身に着ける．

**到達目標**（修得する能力）

- ポインタ，構造体，データ構造，適切な関数呼び出しを使ったプログラムの構成，ファイル操作など，実践的なプログラミングにおけるトピックを理解する．
- アルゴリズム的な考え方の基礎を理解する．
- 効率を考慮して小規模システムをＣ言語で実装できる．

**キーワード**

ポインタ，構造体，データ構造，プログラムの構成，ファイル操作

**関連する科目**

ICT.P204 ： プログラミング基礎（情報通信）

**履修の条件(知識・技能・履修済科目等)**

"ICT.P204 ： プログラミング基礎（情報通信）"を修得していることを強く推奨する

# 実習環境

- 計算機室に来て課題に取り組むことを推奨しますが、ISOまたはJIS規格準拠のC言語であれば何を用いても構いません。Windows版EclipseはISO非準拠なので非推奨です
- 規格準拠のいかなるC言語でも提出物が正しく動作するならその評価は満点となります。
- 自宅で課題プログラムを作成し提出しても結構です。

# 自宅での実習環境構築

T2Schola上の「参考資料（読まなくてもよい）」というフォルダーにWindowsまたはMac上でC言語実習環境を構築する手順書があるので、自宅で課題作成したい人は参考にして下さい。

# 色々な開発環境

- 研究室や就職後の会社で用いる開発環境は今までの講義で用いたものとはほぼ確実に異なります
- 従って、新しい開発環境を与えられたときにそれを使えるようになることが必要になります
- 本日課題ではEclipse以外でC言語を実行しスクリーンキャプチャを提出する課題があります
- プログラミング発展では推奨する開発環境は特にありません
- 特にMicrosoft Visual StudioはC言語プログラムの問題点を日本語で教えてくれるため英語メッセージが苦手な人には便利かも知れません

**講義資料**

T2Schola で提供する

**参考書**

改訂3版 基本情報技術者 らくらく突破 C言語
技術評論社 2019　（注：基本情報処理技術者試験から最近C言語は消えました）

他にも色々（古い本はlong long型の説明が無かったりするので2010年以降に出版されたものを勧めます）.

**授業の進め方**

毎回の授業の終わりに課題に対する回答を提出する.

基本的には，授業時間内に提出するが，終わらない場合は，
T2Scholaに示された締め切り（次回講義開始時かその少し前）までに提出する.

提出の仕方は後述する。

**成績評価の基準及び方法**

1) 各回の項目についての理解度を評価する
2) 評価は提出されたレポートによる（100%）
3) 全出席（＝全課題の提出）が原則である.

計算機室に来たいが来られない（発熱等）や期限までに課題提出できない（入院等）ときはすぐに尾形・松本に**スラック**DMして下さい

# プログラミング発展2023の内容（予定）

第1・第2回 プログラミング基礎の復習とアルゴリズムの基礎
第3・第4回　構造体、共用体、プログラミング基礎の復習
第5・第6回 ポインタ
第7・第8回 データ構造
第9回 ファイル入出力１
第10・第11回 CLI、argcとargv、分割コンパイル、
　　ビット演算、移植性等
第12回 ファイル入出力２

それ以降 最終課題ならびに発表（予定）

# 課題の提出方法について

- T2Scholaの各講義回にある課題に提出の仕方（ファイル名やファイルの種類など）が指示されているので、それに従う

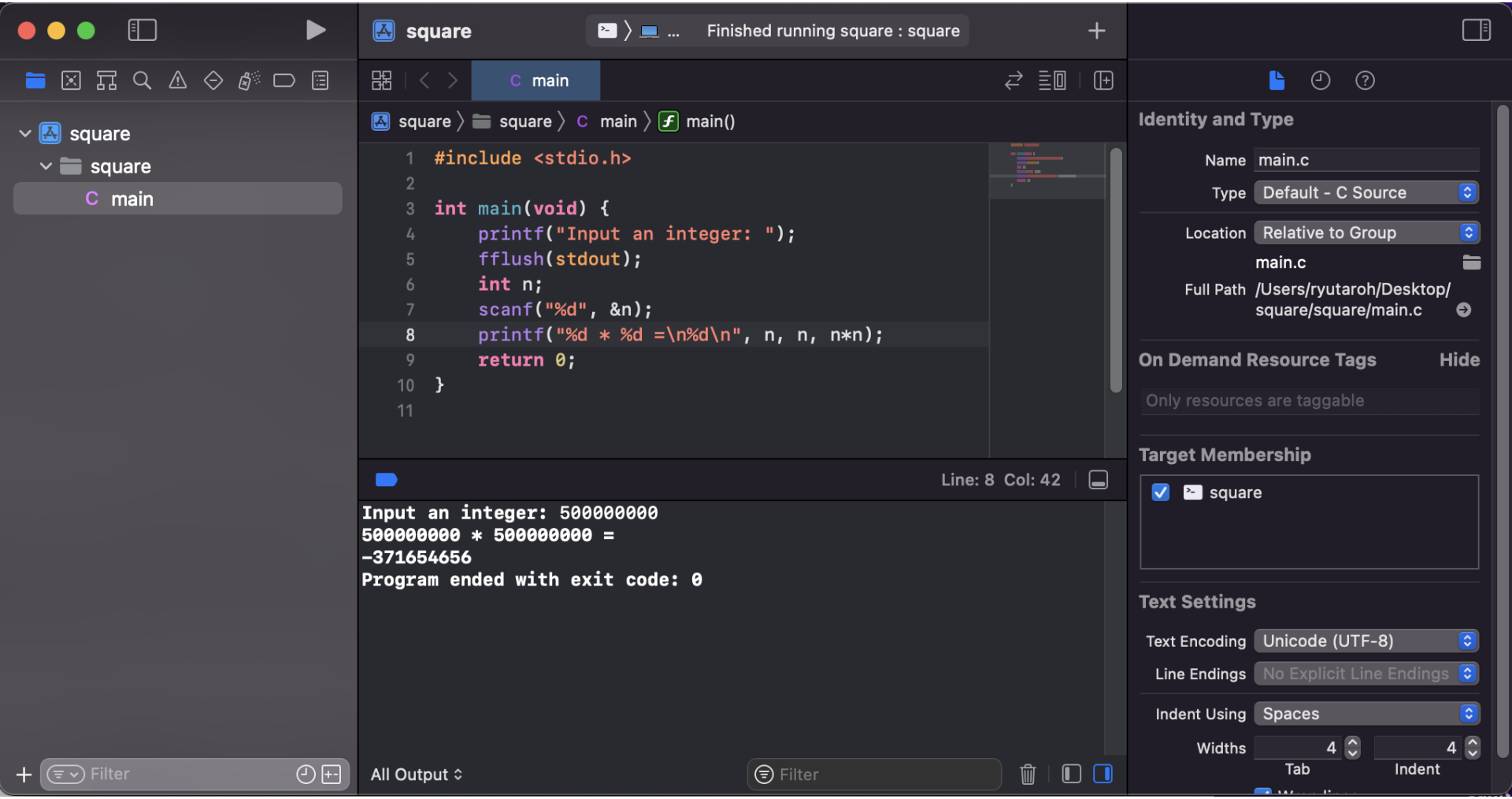
- 各提出物について、簡単なフィードバックをT2Scholaの機能を用いて行うことを計画している　（採点環境においてプログラムが正しく実行できたかどうか、など）

# Ex1-1 (異なる開発環境の使用)

1. 整数型変数にscanf を用いて数値を格納し読み取った数値の二乗をprintfで出力するプログラムを作成し
2. それを**Eclipse以外でコンパイル**（実行可能ファイルの作成）し、
3. プログラムソースと1000000000をキーボードから入力したときの実行結果の両方が写されていて**Eclipseが写っていない**スクリーンキャプチャをPNG形式の画像で提出せよ。

- 次ページのような画像が解答として期待されているが、次ページ画像は入力数値が 5 億なのでそのまま出すと不正解。
- **次ページのように二乗の結果が明らかにおかしくてもきにせず提出してよい。**
- 計算機室のEclipse以外の開発環境には「Xcode」と「ターミナル」がある。自宅のPCでEclipse以外を使用してもよい。
- 提出ファイル ex1_1.png
- 計算機室で既にインストールされているXcode使用方法はT2Scholaの「参考資料（読まなくてよい）」にある。講義スライド説明後にXcodeをデモする。

# Ex1-1 Xcodeによる誤答例



# C言語の型

課題1-1誤答例において結果がおかしかった原因は、計算結果がint型で表現できる範囲に収まらなかったからである。

- 整数型
  char ≦ short ≦ int ≦ long ≦ long long
- 浮動小数点型
  float ≦ double ≦ long double
- 上記で≦は表現できる数の範囲の大小を示す

# C言語整数型の表現範囲（典型例）

| 型の名前 | 最小値 | 最大値 |
|---|---|---|
| char | -128 | 127 |
| short | -32768 | 32767 |
| int | －２０億くらい | ＋２０億くらい |
| long | int**または**long longと同じ | int**または**long longと同じ |
| long long | －(20億の二乗)くらい | +(２０億の二乗)くらい |

- int型が小さすぎるときにlongにしても何も症状が変わらないことがある（たとえばウィンドウズでは long==intであるため）。従ってint型の対応範囲が狭すぎるときには**long long型**に替えるとよい。
- 型ごとに表現範囲が異なるのは、整数を表現するために用いている（二進数としての）**桁数**（ビット数）**が型ごとに異なる**ため。

# 適切な型の選択

- 用途に応じて適切に変数型を選択できる必要があるが、
- 各型で扱える範囲は例えばMacOSとWindowsでは異なる！！
- 必要に応じて、各変数型で扱える範囲を調べられるようになることが必要である。
- 調べる手段としては、書籍や開発環境のオンラインヘルプ、インターネット検索など。

# printfとscanfの型指定文字

- printf で数値を表示するとき、およびscanfで数値を読み取るときに数値の型を正しく指定しないと表示・入力がおかしくなることがあるため、intとfloatとdouble型以外の引数には必ず型指定文字を付ける
- 例として、数値がunsigned charまたは signed charであるときにはhhをつけ、unsigned longまたはlongであるときにはlを付ける（例えば%ldや%luや%lx）。
- これらは1999年のC言語標準規格で定められた
- **型指定文字が正しく付いていない提出物は減点する可能性がある**
- %の後の型指定文字についてはウィキペディアに説明がある https://ja.wikipedia.org/wiki/Printf#%E9%95%B7%E3%81%95%E4%BF%AE%E9%A3%BE%E5%AD%90

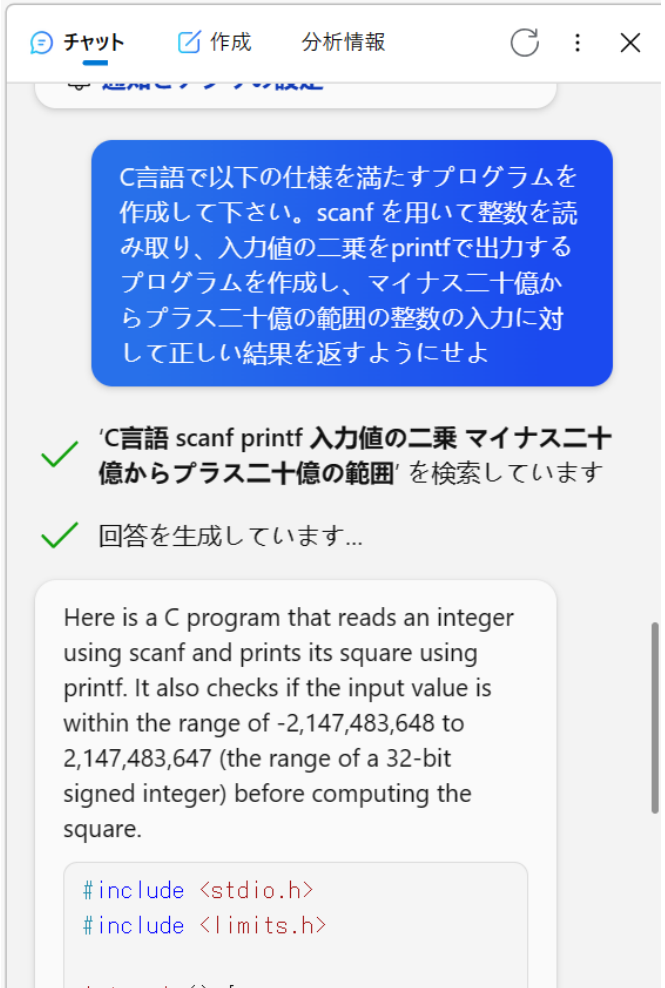
# Ex1-2(変数型の選択)

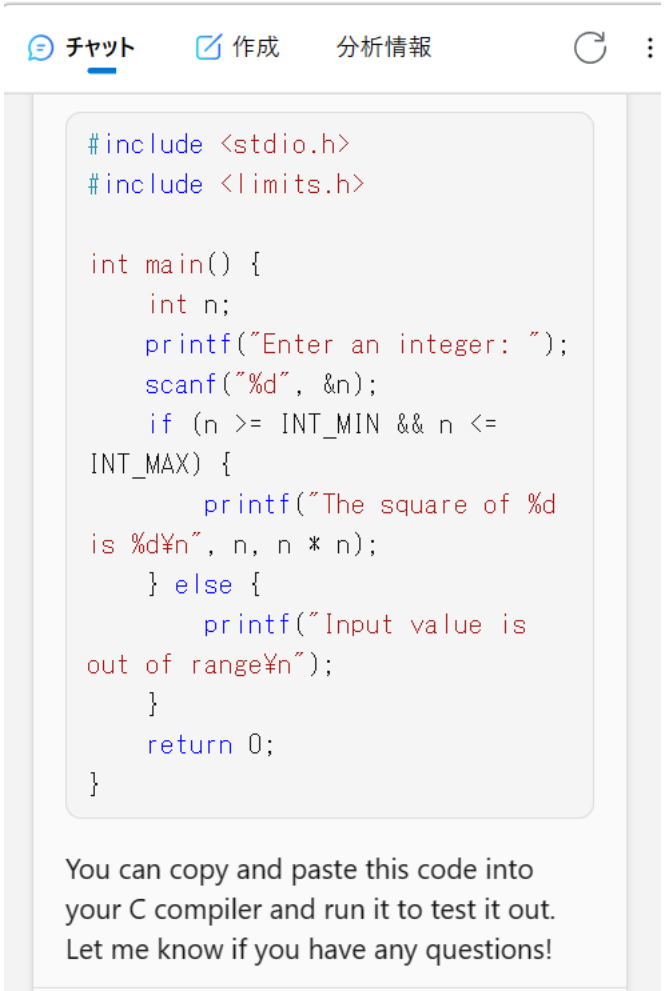
1. scanf を用いて整数を読み取り、入力値の二乗をprintfで出力するプログラムを作成し、マイナス二十億からプラス二十億の範囲の整数の入力に対して正しい結果を返すようにせよ
2. **特に−1000000001（マイナス十億一）を入力した結果が正しいことを確認せよ**

- 結果だけが最後の行に表示されるようにすること、例えばprintf(“¥n%d¥n” kekka) などとすること
- 計算機室の環境(EclipseまたはXcodeのどちらか)で正しく動作しないと減点となる。MacOS上のEclipseとXcodeで動作の違いはない
- 提出ファイルex1_2.c

# Ex1-3(AIで解答できるのか)

Bing AIに課題1-2を答えさせた以下のプログラムのおかしい点をなるべく多く指摘せよ（松本は2つ気付いた）。提出ファイルex1_3.pdf



# チャットAIに関する扱い

禁止も推奨もしない。その理由は、

- 入力やAIの種類に依存して、適切な応答も不適切な応答もあり、
- 応答を精査して用いる必要がある（課題１－３参照）。
- しかし、AI応答の妥当性を精査できる学生もいればできない学生もいるため、一律に方針を定めることが困難であるため。
- （なお同じことはグーグル検索の検索結果にも当てはまる）

# アルゴリズム的考え方の基礎１

・探索
・ソーティング

いろいろな方法（アルゴリズム）が存在

Q どちらのアルゴリズムがより良いのか？

# アルゴリズム的考え方の基礎１

・探索
・ソーティング

⇨ いろいろな方法（アルゴリズム）が存在

Q どちらのアルゴリズムがより良いのか？

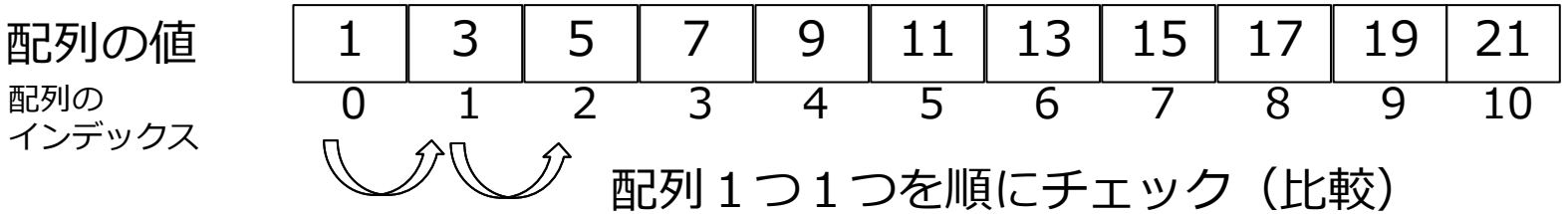
A - 実際に（実行時間や使用メモリ量などを）計測して評価
- 対象となるデータ量nが十分に大きい時に
最良，最悪，平均の計算量により評価

最良：ラッキーヒット的な場合が多いので、
評価としてはあまり用いられない
最悪・平均：一般的

# アルゴリズム的考え方の基礎１

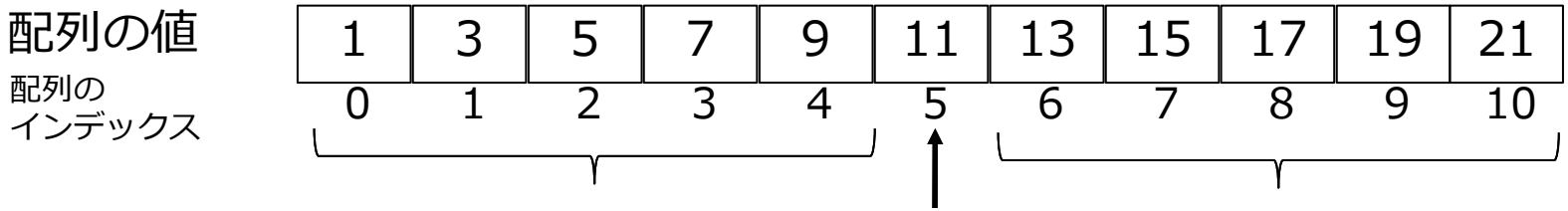
・探索（復習）

- 単純（線形）探索




配列１つ１つを順にチェック（比較）
＊ このように並んでいなくても適用可能

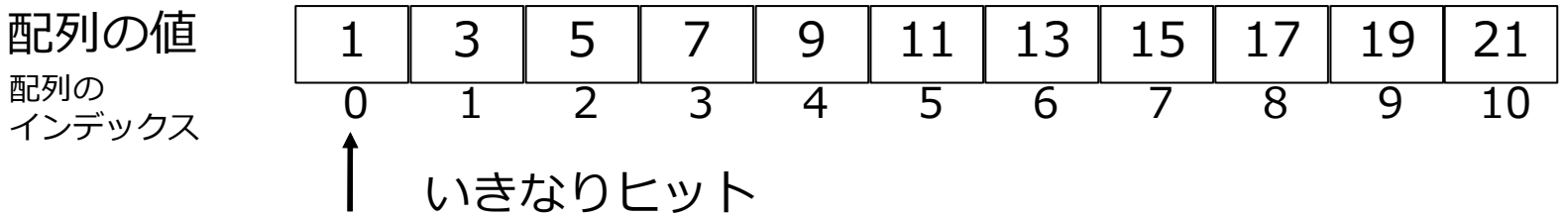
- 二分探索



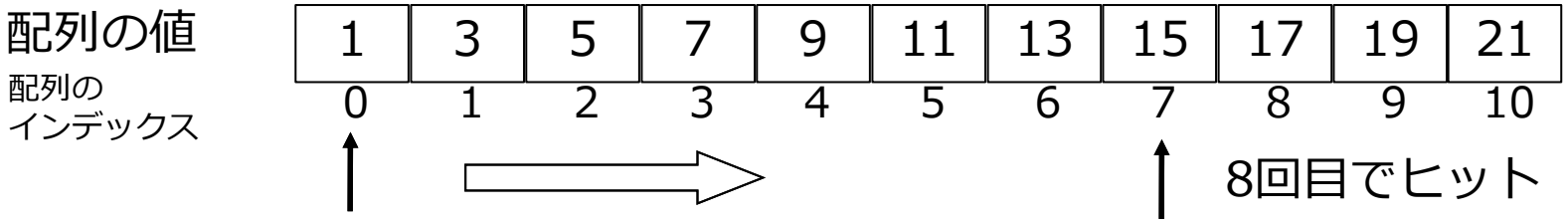

配列を半分の領域に分けながらチェック（比較）
＊性質上、**順番に並んで**いないと適用できない

# - 単純探索

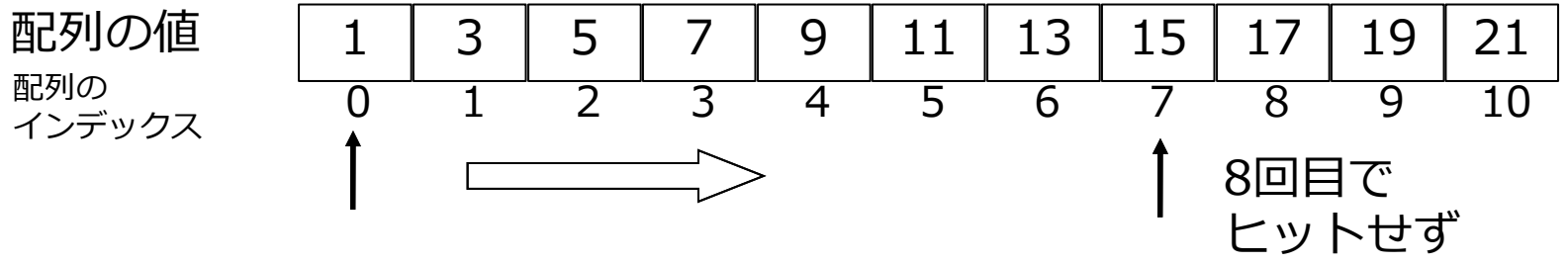
key=1




key=15




key=14



- 単純探索

データの数 “*n*”が増えていったとき,
アルゴリズム（プログラム）の実行にかかる時間は
どうなるか？・比較の総数で測ることにする

仮定：＊keyはデータ列の中にある数（簡単のため）
（Exではデータ列外のkeyも取り扱う）
＊keyとデータ列中の数との比較を１回とカウント
（ Exでは比較が連続(if文が連続)している場合を
“一連の比較“として”比較１回“とカウントする）
（一連の比較の時間はほぼ同じ（大きく違わない）と考える）

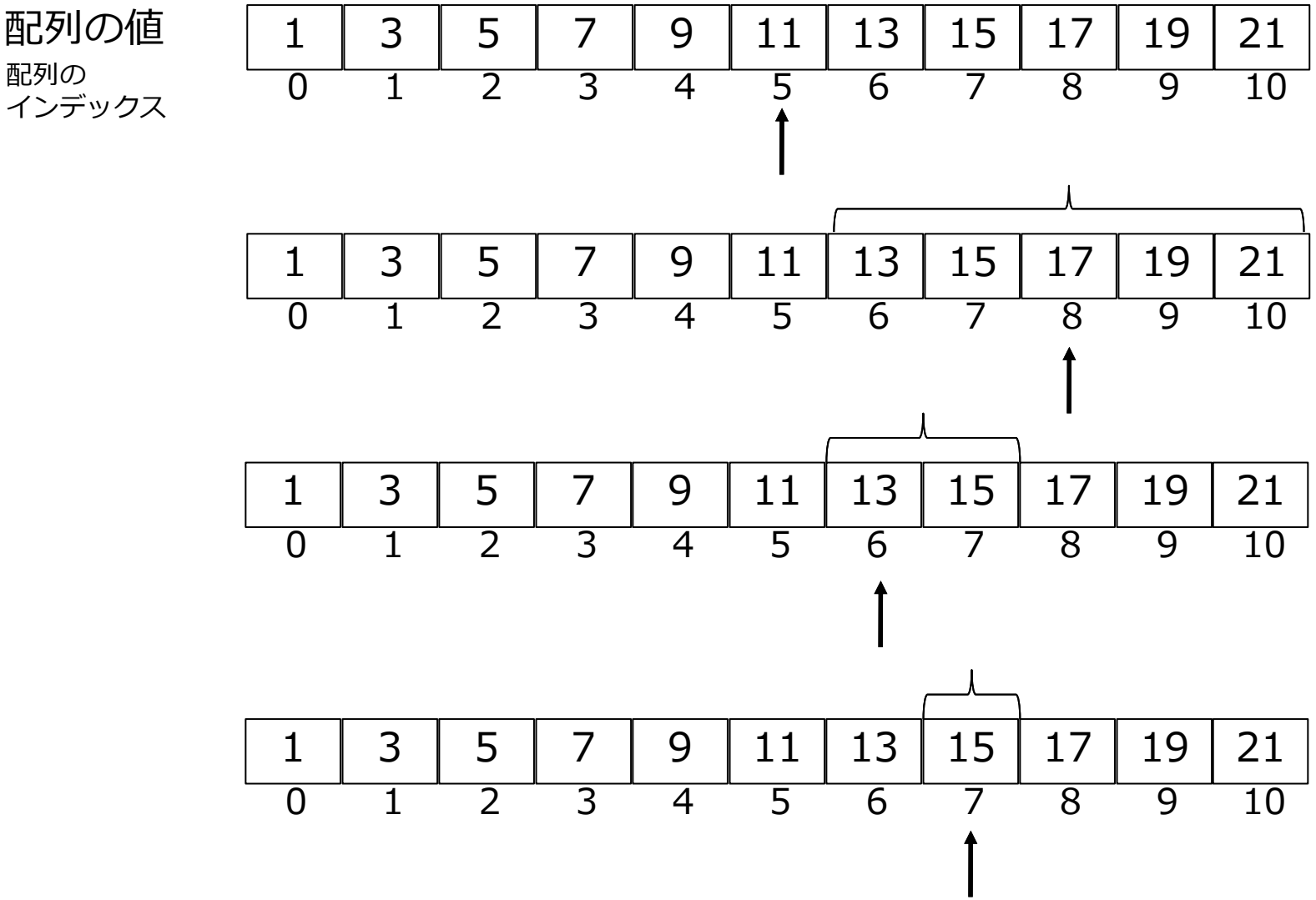
最悪の比較総数は，*n*回

⇨ 最悪の比較総数は*n*に比例・最悪の実行時間も*n*に比例

“O(*n*)”（オーダー*n*）(漸近的時間計算量)と表記します

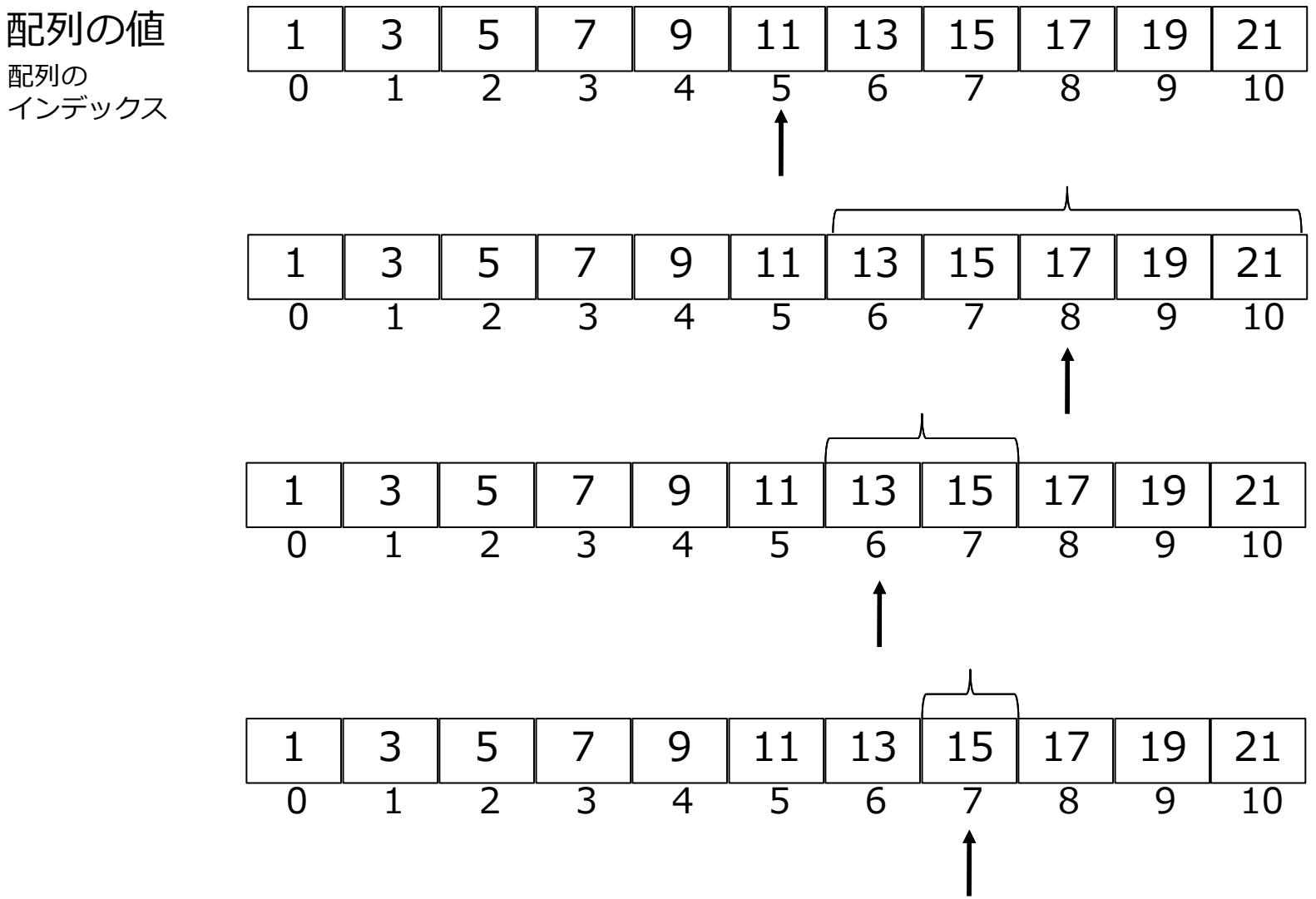
# - 二分探索

最長の比較回数で見つかるとき (key=15)



# - 二分探索

(最長の比較回数で)見つからない (key=14)



- 二分探索

データの数 “*n*”が増えていったとき,
アルゴリズム（プログラム）の実行にかかる時間は
どうなるか？・比較の総数で測ることにする

仮定：＊keyはデータの中にある数（簡単のため）
（Exではデータ外のkeyも取り扱う）
＊keyとデータ列中の数との比較を１回とカウント
（ Exでは比較が連続(if文が連続)している場合を
“一連の比較“として”比較１回“とカウントする
（一連の比較の時間はほぼ同じ（大きく違わない）と考える
＊*n*は十分大きいとする

１回比較後、対象となるデータ数は $\frac{1}{2}n$

２回比較後、対象となるデータ数は $\left(\frac{1}{2}\right)^2 n$

：

*k* 回比較後、対象となるデータ数は $\left(\frac{1}{2}\right)^k n$

- 二分探索

データの数 “$n$”が増えていったとき,
アルゴリズム（プログラム）の実行にかかる時間は
どうなるか？・比較の総数で測ることにする(続き)

最悪の比較回数= $k$ 回比較後、対象となるデータ数が1

$$\left(\frac{1}{2}\right)^k n = 1 \quad \Longrightarrow \quad k = \log_2 n$$

＊ 厳密には$k$は$\log_2 n$を下回らない最小の
正整数（小数点以下切り上げ;ceil($\log_2 n$)）

⇨ 最悪の比較総数は$\log n$に比例
・最悪の実行時間も$\log n$に比例

“O($\log n$)”（オーダー$\log n$ ） (漸近的時間計算量)
と表記します

ちなみに $n > \log n$ （$n > 0$）

同様の議論の結果，平均の実行時間も “O($\log n$)”

＊ 導出はちょっと大変なので省略

# Ex1-4(探索の計算量)

プログラミング基礎Ex35

下記の100の要素からなる配列に対して，0から127までの整数を探索のキーとして128回の2分探索を行い，各キーの配列における位置，各キーを見つけるまでの探索（比較)回数，その平均探索(比較)回数を出力するプログラムを作成せよ.

```
int data[100] = {
    0, 1, 2, 4, 5, 6, 7, 9,10,12,13,15,16,17,18,19,20,21,
   23,26,27,28,29,30,31,33,34,35,37,38,39,40,41,42,44,46,
   47,48,49,51,52,53,54,55,56,57,59,60,61,63,64,65,67,68,
   71,72,73,74,75,76,77,78,79,80,83,84,85,86,87,89,90,91,
   92,93,94,95,96,97,98,99,101,102,104,105,107,108,110,
   111,112,113,114,115,117,118,119,120,123,124,125,127
};
```

* 比較対象が探索で見つからないときは，“見つからなかった”(“not found”)とする.

を思い出して下さい.

これについて，以下の変更を加える.

（元のプログラムは残しておくこと）

# Ex1-4 （続き）

1） Ex.35の平均探索（比較*）回数をプログラムにより求めよ（再）
＊ 二分探索における 連続する比較を１回とカウントする

★提出物：特になし
（平均探索（比較）回数の出力は提出する必要はないが.
数値は５）で必要になります）

2）Ex.35の100の要素からなる配列に対して，0から127までの整数を探索のキーとして128回の**線形**探索を行い，各キーの配列における位置，各キーを見つけるまでの探索(比較)回数を求め，
その平均探索 (比較)回数を出力するプログラムを作成せよ.

★提出物：プログラム ( １つのファイルにする）

“ex1_4_2.c”
＊ 実行すれば，平均探索 (比較)回数が出力されるようにしておく

（平均探索（比較）回数の出力は提出する必要はないが.
数値は５）で必要になります）

# Ex1-4 （続き,小問3は欠番）

４）Ex35のデータ列の要素を0から255までの数値200個に伸長して（元のデータ列に128を加えた数を追加する），0～255までの整数を探索のキーとして合計256回の探索を行うように，１～２）のプログラムをそれぞれを書き換えて，平均探索（比較）回数を計測せよ.

即ち，作成するプログラムは，要素数が200のデータ列に対して，

・平均探索（比較）回数を計測するプログラム

線形探索及び二分探索

の都合２種類を用意することになる.

★提出物：200個のデータについての平均探索(比較)回数を出力するプログラム

二分探索　　“ex1_4_4.c”

＊ 実行すれば，平均探索 (比較)回数が出力されるようにしておく

その他についてのプログラムは提出は求めない

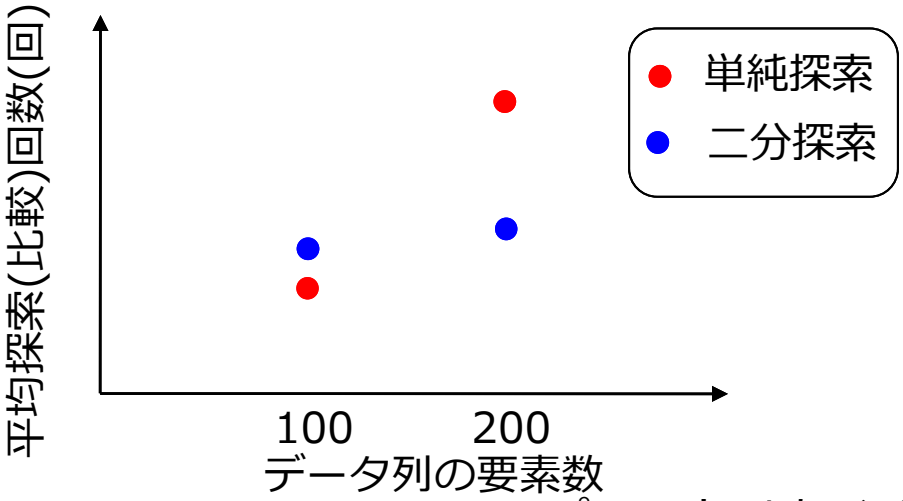
（平均探索(比較)回数を提出する必要はないが，数値は５）で必要になります．）

# Ex1-4 （続き）

5） 1)～4)で得られた平均探索（比較）回数を対比してみる．それぞれについて，表にしてみよ．また，横軸をデータ列の要素数，縦軸をそれぞれの数値（平均探索回数）として，グラフにプロットして整理してみよ．

平均探索(比較)回数(回)

| データ列の要素数 | 100 | 200 | |
|---|---|---|---|
| 単純探索 | | | |
| 二分探索 | | | |




プロットはあくまで例です。

# Ex1-4 （続き）

５）（続き）
★提出物：平均探索回数の結果をまとめた表及び
　　プロットしたグラフ　（提出は 6）と一緒のファイルに）

６）５）の表及びプロットから，線形探索と二分探索の計算量について考察し，対象のデータ列の要素数が400,
800… と増えていったとき，探索の平均回数がどうなると考えるか予想せよ.

★提出物：５）の表とグラフ
　　及び
　６）の考察
　　をワープロソフト等を利用して１つのファイルにまとめ,
　　**pdf**に変換して提出

“ex1_4_6.pdf”

# Ex1-4 （続き）

７）★発展的課題（余力のある人向け)
４）で探索対象のデータ列の要素数を300，400…と増加した場合に，線形探索及び二分探索それぞれについて平均探索(比較) 回数を計測するプログラムをそれぞれ作成してみよ

★提出物
計測結果を，5) の表及びプロットに反映した場合は
加点対象とします

# Ex1-4 （ヒント）

4)
データ列の要素数を伸長するには,

元のデータ列を data[] ・ data_orig[] に変更した上で,

```
int i;

    for (i=0;i<100;i++) {
        data[i]=data_orig[i];
        data[i+100]=data_orig[i]+128;
    }
```

などとする.

- data[] の宣言を要素数に合わせる
- 探索の範囲を変更する

ことを忘れずに

# 悪いプログラムの例

- main() 関数にすべての処理が書いてあり、3000行あるプログラム
- 改行が少なすぎて１行に１０００文字書いてあるプログラム
- Windowsでは動作するがMacでは動かないプログラム、Intel CPUでは動くがM1 MacやスマホのARM CPUでは動かないプログラム等等

# 巨大関数の何が悪いのか

- 処理の流れが追えないため、他人や将来の自分が変更を加えることが困難になるから
- 機能ごとに関数に分けられず全部main()にあると再利用できないから
- 採点者が提出されたプログラムを理解することが困難で部分点を与えられないから

# １行に１０００文字あると何が悪いのか

- エディターによってははみ出して行全体が見えないから
- はみ出さない場合、変なところで改行されて読みにくいから

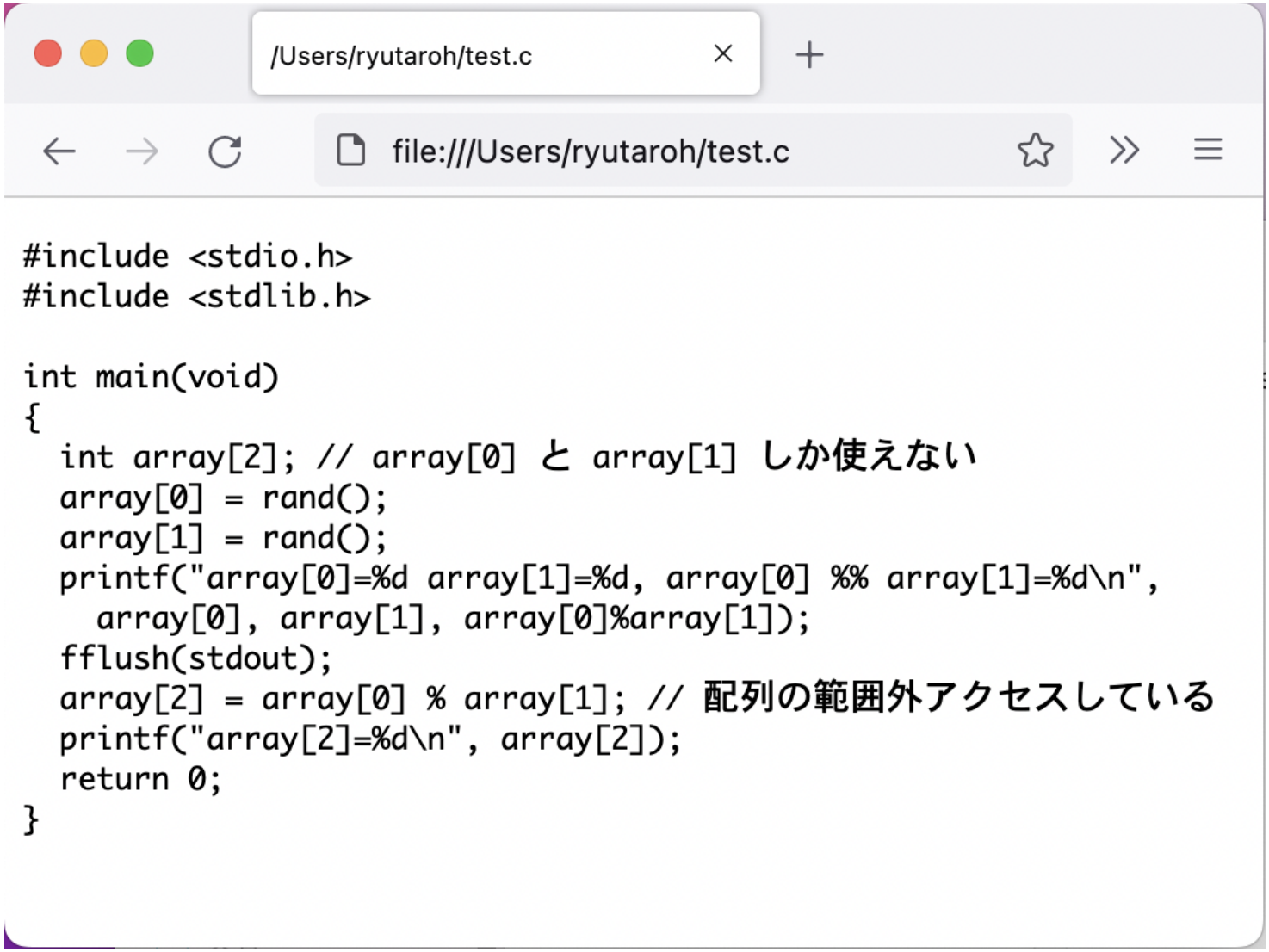
# 他の環境で動かないプログラムの何が悪いのか

- この講義の採点者が動作確認できず採点できない（**採点はLinuxで行う**）
- プログラムを動作させる機材やOS(Windows等)をより新しくより便利なものに置き換えられなくなる
- 例：みずほ銀行の法人口座のオンラインバンキングはInternet Explolerでしか動作しない(2022年になったら治った模様)

# 未定義動作

- 配列の範囲外アクセスや、未初期化変数からの読み出しを行うと、C言語規格では**動作が未定義**で「何が起きるか全く保証できない（異常終了するかもしれない）」と書いてあります。
- 自動採点システムではこれら未定義動作を検出し、未定義動作はすべて減点します。

# 未定義動作を含む例



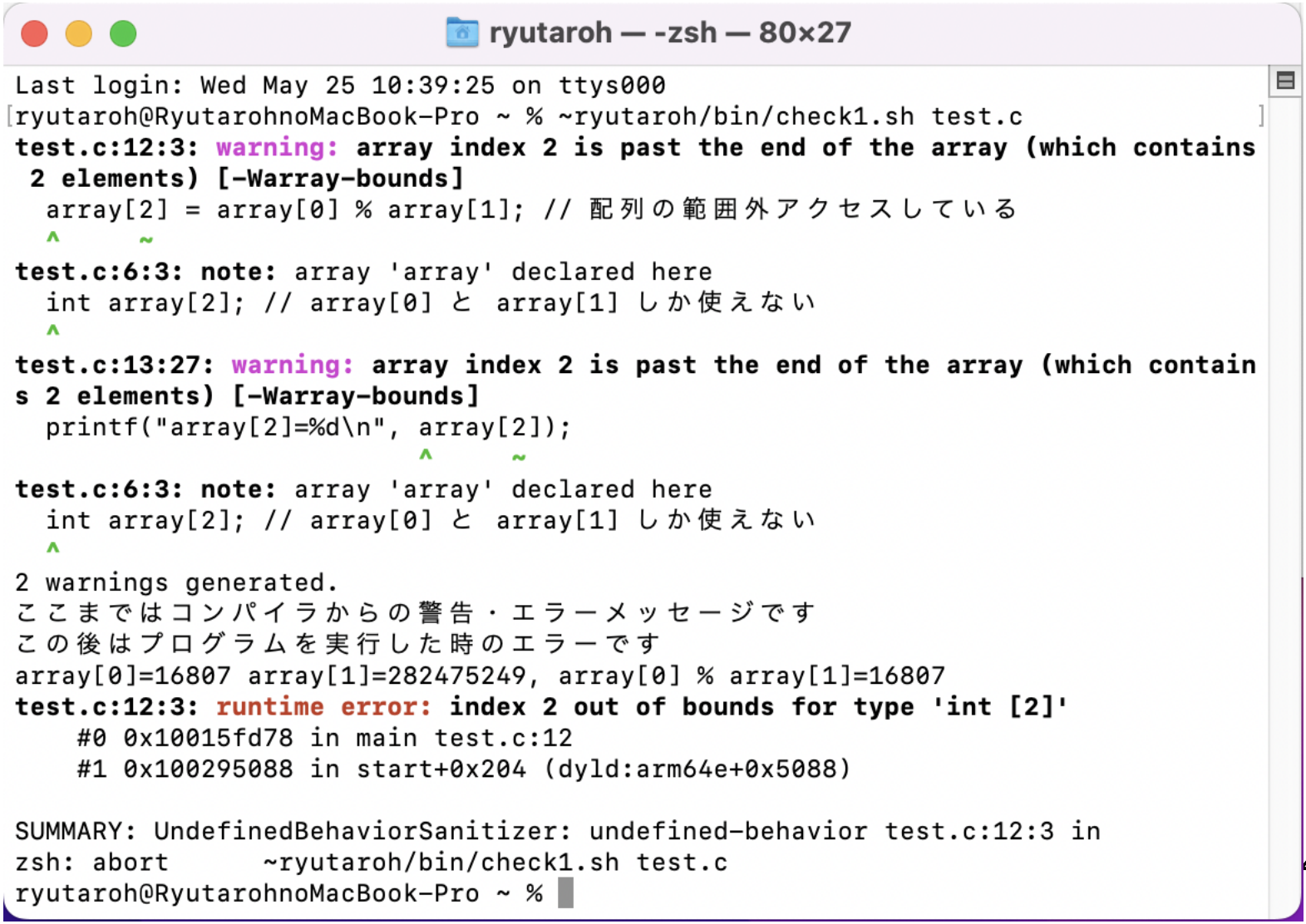

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  int array[2]; // array[0] と array[1] しか使えない
  array[0] = rand();
  array[1] = rand();
  printf("array[0]=%d array[1]=%d, array[0] %% array[1]=%d\n",
    array[0], array[1], array[0]%array[1]);
  fflush(stdout);
  array[2] = array[0] % array[1]; // 配列の範囲外アクセスしている
  printf("array[2]=%d\n", array[2]);
  return 0;
}
```

# MacOSターミナルの起動

未定義動作を検出するための手順を紹介する。まずターミナルを起動する

# 未定義動作を調べる検査

ターミナルから ~matsumoto.r.aa/bin/check1.sh ナントカ.c を実行して未定義動作を見つけることができる



# printfとfflushによるデバッグ

- プログラムが意図したとおり動作しないとき、その変数が意図しない値になっていることが多い
- したがって、意図通り動作しない場合、怪しい部分で関連する変数をすべてprintf等で表示すると状況を把握できることが多い
- printf**の後で**実行時エラーが起きているにも関わらず、printf内容が表示されないことがある
- 上記も含めてprintf内容が表示されない場合printf直後にfflush(stdout);を置くと、確実に表示が行われる
- 表示されない理由は後の回で説明するが、気になる人は「バッファリング printf デバッグ」などで検索すると関連情報が得られる

# うまく動かない質問時のお願い

- プログラムが上手く動作していないと疑わしい部分の変数を printf で表示するようにしてから質問して下さい（課題提出時にはデバッグ用printfは消す）
- （講師が松本のときのみ）質問する前に「check1.sh」によるコンパイラの警告と実行時エラーを無くして下さい。警告やエラーの意味がわからないときは気軽に聞いてください

# Ex1-5(次回への準備)

1）プログラミング基礎の「文字」と「文字列」、高校の「情報」ならびに「数学」にあったｎ進数を思い出して下さい。次回課題で使います。
2)プログラミング基礎で学習した

- 単純なソーティング方法：選択法（資料中で紹介）
- 単純なソーティング方法：交換法(バブルソート；Ex39の答え),
- 高速なソーティング方法：クイックソート(Ex41),
- 高速なソーティング方法：マージソート(Ex42)

のプログラムを全て完成しておいて下さい.

例えば，以下の数値列の例に対して，ソート結果が正しく出力されることを確認して下さい.

{ 0.0，1.4，5.3，7.8，10.2，3.5，5.3，8.9，2.6，7.8 }

また，ソートの対象となる任意の数値列をランダムに生成する方法についても復習しておいて下さい.
★提出する必要はありません。次回、これらをプログラムで利用します.
★次回資料がT2Scholaにあるので時間が余ったら予習と解答をお願いします

# 提出ファイルの一覧

- ex1_1.png
- ex1_2.c
- ex1_3.pdf
- ex1_4_2.c
- ex1_4_4.c
- ex1_4_6.pdf

# コピペレポートについて

プログラムや考察などが他の提出者と重複している場合、不正とみなして減点および問い合わせをすることがあります。
減点をする場合、コピー元とコピー先の区別が付かないため類似性が高いレポート提出者全員を減点します。

# コピペの判定基準

- 提出物のすべての対（つい）に対して、類似度を求める
- 得られた値の平均と分散を計算する
- 分散と平均がわかると（予備校の模試などで用いる）偏差値を求められる（確率と統計の講義を復習して下さい）
- 類似度の偏差値が、閾値よりも高いときにコピペと見なす
- この説明でうまくいく理由を理解することはある意味で確率と統計の履修範囲…