

プログラミング発展

2023年度2Q 火曜日5~7時限(14:20~16:50)
金曜日5~7時限(14:20~16:50)

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

(最終更新 : 7月16日16 : 00)

最終課題

- 情報通信概論で説明されたRSA公開鍵暗号を扱う
- 今までの講義内容の多くを再び使う（次ページ参照）
- RSA暗号自体は既に情報通信概論で既習なので説明を繰り返さないが、わからないことがあれば説明するので、気軽に相談して欲しい

課題で必要な要素

- 構造体と整数型が扱える数値の範囲（1, 3 回目）
- ポインタ（5 回目）
- free()（とmalloc()）（6 回目）
- ASCIIコード表とファイル（9 回目）
- ビット演算（とエンディアン）（10 回目）
- 分割コンパイル, argv[]（12 回目）
- 効率的べき乗演算（情報通信概論）

復習に時間が掛かるかも...

- 前のページで列挙した項目の一部があやふやだったり忘れたりしていると、その部分で引っ掛かります
- どの部分の学習が抜けているか自分でわからない場合には相談して下さい
- 頭の中から抜けている項目を見つけたら、面倒がらずに過去講義資料を復習するか、C言語の参考書などでその項目を学習しておして下さい

無限精度の整数

- RSA暗号では桁数のとても大きい非負整数を扱うが、大きすぎてC言語の型では扱えない
 - 参考：工学リテラシで出てきたPythonの整数は無限精度で、いくらでも大きい整数を扱える。従ってPythonの整数に桁あふれはない
- 無限精度の整数演算そのものを最終課題にするのは難易度が不適切なので、無限精度の整数演算を行うための構造体と関数を与える

無限精度の非負整数を扱う型

hugeint.hで定義されている

```
typedef struct {  
    int size;  
    unsigned char num[];  
} huge_int;
```

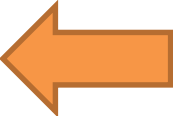
- マイナスの整数は表せない
- num[0]~num[size-1] までに**リトルエンディアン**で符号なし整数が**256進数**で格納される
- size == 0 の場合は値がゼロであると見なす
- size < 0 の引数を下記関数群に渡すとエラーになる。
- 上記の整数型に値を設定する手順はhugeint.cのfrom_uchar()を参考のこと

huge_intの値と配列numの関係

- 配列numに256進数で非負整数 x の値が格納される
- 256進数の各桁を配列numに並べる順番はリトルエンディアンである。従って
- $x == \text{num}[0] + 256 * \text{num}[1] + 65536 * \text{num}[2] + \dots + 256^{(\text{size}-1)} * \text{num}[\text{size}-1]$ という関係が成り立つ
- $\text{size} == 8$ のときには課題10-4におけるucとullの関係と同一である

長さ指定しない配列

- 構造体の要素として現れる配列要素は、その配列要素が構造体の最後にある場合に限って、長さ指定を省略できる

```
typedef struct {  
    int size;  
    unsigned char num[];   
} huge_int;
```


長さ指定しない配列を持つ構造体

```
typedef struct {  
    int size;  
    unsigned char num[];  
} huge_int;
```

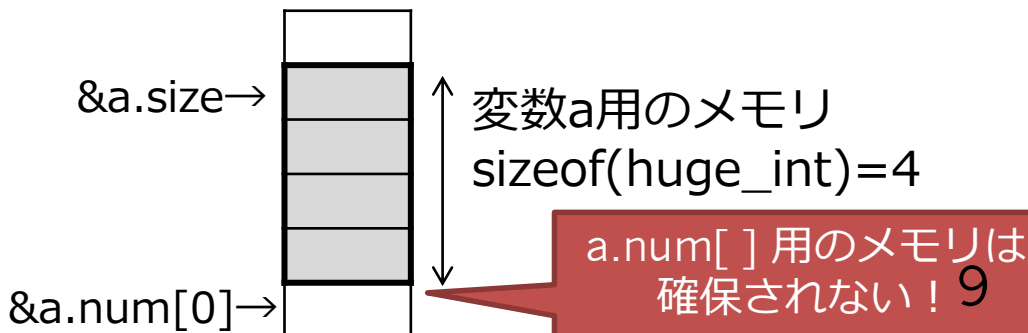
構造体の大きさ (sizeof) は**最後の配列要素が無い場合と同じになる**

→構造体の変数を**普通に宣言すると、配列用のメモリは確保されない**

例 :

huge_int a;

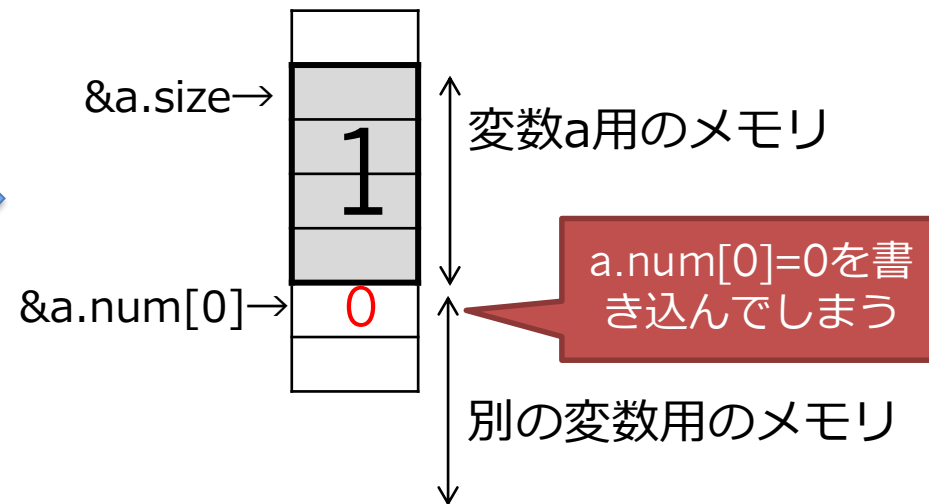
→



長さ指定しない配列を持つ構造体

- 配列に値を代入しようとするとなくなるか

```
huge_int a;  
a.size = 1;  
a.num[0] = 0;
```



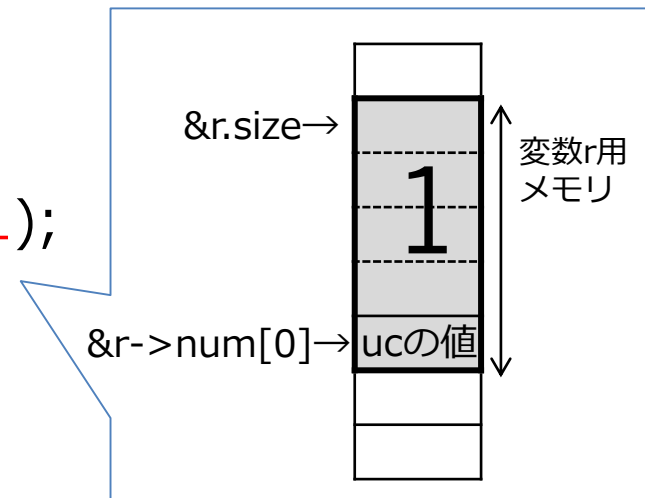
- 構造体aの次のアドレスに格納された、関係ない変数の値が破壊される

長さ指定しない配列を持つ構造体

- そのような構造体は**普通に宣言せず、「構造体へのポインタ」を変数として宣言**して、長さ指定がない配列に代入する要素数分のバイト数を、構造体の大きさ(sizeof)に足した長さをmalloc()し、malloc()から帰ってきたポインタを構造体のポインタ変数に代入して用いる（あとで使用例を紹介する）
- 無限精度整数の桁数は幾らでも大きくなり得るため、この書き方がもっともすっきりしている（と思います）
- 自分はもっと上手くできると思う人は、最後の課題に挑戦して欲しい
- huge_int型および関連関数を独自改良して使う場合、異なる名前を付けて提出物に定義を含めること。課題の要求を満たせばhuge_intをそのまま使うか否かは自由である。内容を変更しないのに名前だけ変えて提出物に含めると減点。変更して使う場合必ず変更内容を課題14-7提出物で説明すること

huge_int型への値の設定（例）

```
// hugeint.c より抜粋
// 普通のunsigned charからhuge_int型に変換する
huge_int *from_uchar(unsigned char uc)
{
    if (uc > 0) {
        huge_int *r = malloc(sizeof(huge_int)+1);
        r->size=1; r->num[0]=uc;
        return r;
    } else {
        huge_int *r = malloc(sizeof(huge_int));
        r->size=0;
        return r;
    }
}
```



uc=0の時でも, size=1
として構わないが、
1バイトメモリが無駄に
なる

利用できる関数：普通の整数型の値を huge_int型に変換するための関数

// hugeint.c より抜粋。constはポインタで指される内容を書き替えないことを**この文脈**ではしめす

// 以下のhuge_int *ポインタを返す関数群はすべてmalloc()で確保したメモリ領域へのポインタを返すため、使わなくなった後に返り値のポインタをfree()しないとメモリリークでサニタイザに**怒られる**

- huge_int *from_uchar(unsigned char uc); // 普通のunsigned charからhuge_int型に変換する
- unsigned long long to_ulonglong(const huge_int *x); // 普通の整数型に変換する。x->size >= 9のとき実行時エラーを引き起こします

利用できる関数：huge_int型の整数を演算するための関数 1

- `bool is_zero(const huge_int *x);` // `x==0`のときだけ真になる
- `huge_int *huge_add(const huge_int *a, const huge_int *b);`
// 足し算
- `huge_int *huge_subtract(const huge_int *a, const huge_int *b);` // 引き算。結果が負ならNULLを返す
- `huge_int *huge_multiply(const huge_int *a, const huge_int *b);` // 掛け算

利用できる関数：huge_int型の整数を演算するための関数 2

// 以下は割り算の剰余と商を格納するためのデータ型である
typedef struct {

 huge_int *q; // 割り算の商 (quotient)
 huge_int *r; // 割り算の剰余 (remainder)
} huge_pair;

//以下はaをbで割った商と剰余を返す

- huge_pair huge_divide(const huge_int *a, const huge_int *b);

//上記の関数群の使い方は素数を1000個探して表示する
sieve.c を読んで参考にして欲しい

sieve.c は、値がlonglongで手におえる範囲でしか動作しないです

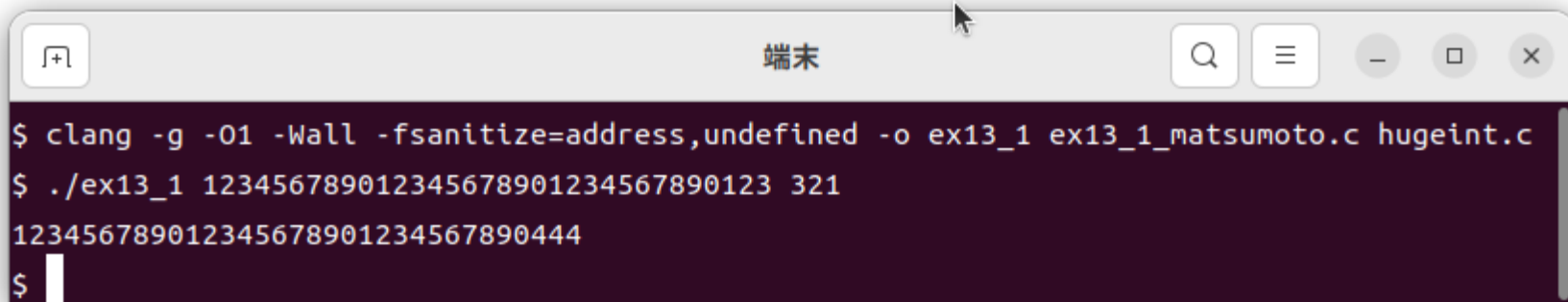
メモリの解放について

- huge_int型のメモリをプログラム終了時に漏れなく解放することはかなり煩雑である
- サンプルの sieve.c を見るとその煩雑さがわかる
- 最終課題については、プログラム終了時のfree()忘れの減点度合いを少なくする

Ex13-1

～huge_intに慣れる～

- 2つの十進数が引数argv[1]とargv[2]として与えられるとする
- それらの和を標準出力に十進数として書き出せ（実行例は以下）
- 入力の十進数は数字の1桁以上の並びとする
- 入力の十進数の桁数がどれだけ長くても正しい結果を表示できるようにする。**特に30桁以上の整数を与えて動作確認すること**。提出物: **ex13_1.c**
- 課題2-1の考え方をargv[i]からhuge_int整数への変換に流用可能
- 課題3-1の考え方をhuge_int整数から十進数文字列への変換に流用可能
- 締め切りは次回講義開始時とし、解答例解説を次回講義にて行う
- argv[1]に引数を与えて実行する方法が使っている統合開発環境でわからない場合、講義でやり方を説明した「ターミナル」を推奨します



```
$ clang -g -O1 -Wall -fsanitize=address,undefined -o ex13_1 ex13_1_matsumoto.c hugeint.c
$ ./ex13_1 123456789012345678901234567890123 321
123456789012345678901234567890444
$
```

hugeint.cをコピーしたら間違い

- check-final.shによる検査ならびに採点は clang ex13_1.c(等) hugeint.c と分割コンパイルで行うので、このようにコンパイルしたときにエラーが出ないようにする
- hugeint.c を「自分のプログラム.c」にコピーして用いると、「clang 自分のプログラム.c hugeint.c」とコンパイルしたときに必ずエラーになるため、減点対象となる
- Eclipseを用いる場合、hugeint.cならびにhugeint.hをプロジェクトのフォルダーにコピーして使うとよい
- この話がぴんと来ない人は分割コンパイルを復習して欲しい
- **13-1以降13-2を除くすべての提出物でこのことに注意すること**

後の課題で流用できるか

- 以降の課題で流用できる関数を作っておくと、後が楽になる
- 後ろの「課題の関係」のスライドを参照のこと

尾形先生の高速べき乗演剰余演算 のスライドを復習する

- 情報通信概論の講義スライドの一部を復習します

Ex 13-2（高速べき乗計算）

- `argv[1]`で与えられる非負整数を x とし、`argv[2]`で与えられる非負整数を y とする。 x^y を本日説明（復習）した方法（ただし剰余演算（mod N ）を行わない）で効率的に計算し、 **x^y を表示するプログラムをex13_2.c**として提出せよ。 x^y がunsigned long longで表現できる範囲内に収まっているときには必ず正しい結果を出さなければならない。その範囲外的时候には例えば実行したときにエラーなどが生じても減点しない。また`argv[1]`と`argv[2]`はint型で表現できる範囲に収まっていると仮定してよい。
- `argv[1]`, `argv[2]`から整数型変数への変換は`sscanf()`, `atoi()`, `strtol()` などを用いることができるので、使い方をグーグルなどで調べて欲しい。`sscanf()`は尾形先生が説明済み。

Ex13-2つづき

- 最終課題のこの後の提出物に流用できるように、割り算および剰余（余り）演算ならびにmath.hで定義される数学関数を用いなくて課題13-2を解くこと（掛け算は使用可）
- 掛け算の回数が最少になるなら情報通信概論で示した方法以外の方法でも構わない（掛け算の回数が最少になる別解が存在する）

Ex14-1

～ファイルをメッセージに変換する～

(RSA暗号で、メッセージを非負整数 m として扱うための準備 1)

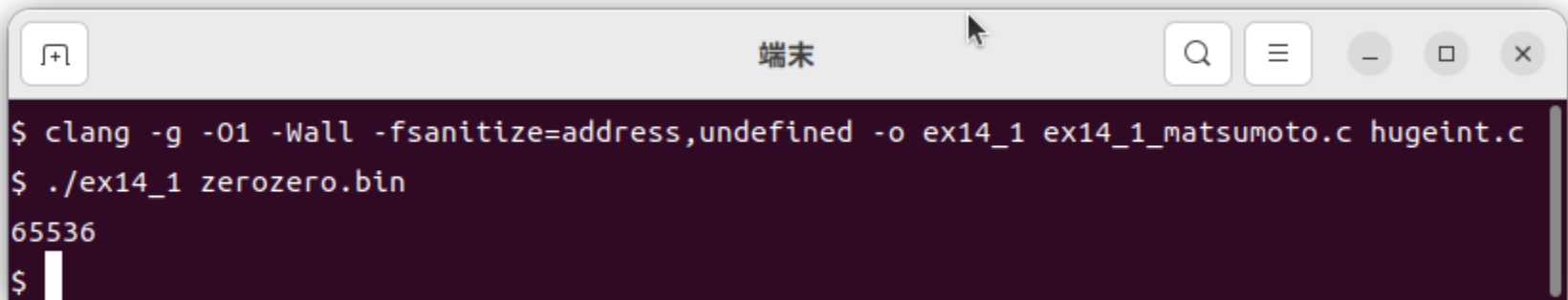
- `argv[1]`でファイル名を受け取り、そのファイルの全内容を、単一の非負整数に変換し`huge_int`型の変数 m に格納した後、標準出力`stdout`に十進数として m を表示するプログラムを作成せよ (m の十進数表現と行末の改行 \n 以外何も出力しないこと) 実行例は次ページ
- ~~ファイルの大きさとして0バイトも有り得る~~
- ファイルがいくら大きくても正しく動作すること
- ファイルの内容と非負整数の対応関係は自分で決める
- **ファイルの大きさが1バイト以上で、内容をメモ帳 (Macならテキストエディット) で正しく表示できる場合には、一対一に非負整数に変換すること。特に漢字を含む場合も考えること。漢字入りファイルがぴんと来ない場合は、漢字入りファイルを以前提出したファイルダンププログラムで表示すること。**
- 提出物: `ex14_1.c`
- 課題 14-1とそれ以降の締め切りは **8月5日 (土) 21時** である
- 解答例解説は8月中にT2Scholaにて公開予定

Ex14-1

～ファイルをメッセージに変換する～

(RSA暗号で、メッセージを非負整数 m として扱うための準備 1)

- 【前頁から再掲】 `argv[1]`でファイル名を受け取り、そのファイルの全内容を、単一の非負整数に変換し`huge_int`型の変数 m に格納した後、標準出力`stdout`に十進数として m を表示するプログラムを作成せよ (m の十進数表現と行末の改行 ¥ n 以外何も出力しないこと) 実行例は



```
端末
$ clang -g -O1 -Wall -fsanitize=address,undefined -o ex14_1 ex14_1_matsumoto.c hugeint.c
$ ./ex14_1 zerozero.bin
65536
$
```


ファイルの大きさを調べる方法

- 課題14-1では、ファイルの大きさ（正確に言えば何バイト読めるか）を知る必要があるかも知れない
- C言語規格に完全準拠したやり方はfgetc()またはfread()でファイル終端に達するまで何バイト読めるか実際に読むしか無さそう
- 動作環境にWindowsやMacを仮定するともっと賢い方法があるが、提出物の中では2018年C言語規格に存在しない関数を用いないこと

Ex14-2

～メッセージをファイルに変換する～

(RSA暗号で、メッセージを非負整数 m として扱うための準備2)

- `argv[1]`でファイル名が与えられ、`argv[2]`に十進数の文字列として非負整数 m が与えられるとき、
- 非負整数 m で表されるファイル内容を、`argv[1]`で与えられたファイルを新たに作りそこに書き込むプログラムを作れ
- 提出物: **ex14_2.c**
- `ex14_1.c` でファイルを十進数に変換し、`ex14_2.c` にその十進数を与えたときには、ファイルの内容が相違なく復元されなければならない

Ex14-3

～メッセージをファイルに変換する～

- 課題14-1と14-2提出物で用いた、ファイルの内容と非負整数 m の対応関係を述べ、その対応関係が一対一だと自分思う理由を説明せよ
- 提出物: [ex14_3.pdf](#)

Ex14-3（発展的課題）

～メッセージをファイルに変換する～

- 課題14-1と14-2において、大きさはゼロバイトのファイルを含めて、あらゆる内容のファイルも一対一に非負整数に変換できる場合には加点を行う
- あらゆるファイルを一対一に非負整数に変換できる場合には、前ページの記述に加えて、以下のempty.bin, zero.bin, zerozero.bin がそれぞれどのような非負整数に変換されるか ex14_3.pdfに明記すること
- make_testfiles.c を実行すると、0バイトのファイル empty.bin、0が1つだけ書いてあるzero.bin、0が2つ書いてあるzerozero.bin ができる。これらを課題14-1と14-2の提出物で扱ったときに、ファイルの元の内容が戻るかどうか確認すること

Ex14-4

～RSA暗号化～

- RSA暗号の平文 m , 公開鍵 $N(=p*q)$, e がそれぞれ十進数文字列で $\text{argv}[1]==m$ $\text{argv}[2]==N$ $\text{argv}[3]==e$ として与えられるとする($m < N$ を仮定して構わない)
- このときに暗号文 c を計算し、標準出力に書き出せ (c の十進数表現と行末の改行 ¥ n 以外何も出力しないこと)
- 提出物: **ex14_4.c**

課題14-4・14-5に関する注意

- 配布物に含まれる **rsa-key-medium.txt**に入っている鍵を用い、最大限の最適化を施して実行可能ファイルを作ったときに、提出物の実行が**90秒**以内に手元の計算機で終わることを確認して下さい
- 課題13-2の考え方を活用してべき乗演算を効率良く行うこと（そうしないと計算が終わらない）
- 最大限の最適化-O3（第12回目講義資料で説明）を用いたほうがいいです（実行時間が短くなるため）
- 使っている統合開発環境で最適化を行う手順がわからない場合、やりかたを講義で説明した「ターミナル」を推奨します
- 実行が終わらない場合自動採点システムでも実行できず、評価が0になる可能性があります
- 90秒以内に終了できるように提出物を作成できない場合、**オフィスアワー8月4日13:40～16:30（計算機室）**に来て相談をして下さい
- あとで公開する解答例の実行時間は、最大限の最適化を行い rsa-key-medium.txt を使用したときに 3 0 秒程度

Ex14-5

～RSA復号～

- RSA暗号の暗号文 c , 公開鍵の一部 $N(=p*q)$, 秘密鍵 d がそれぞれ十進数で $\text{argv}[1]==c$ $\text{argv}[2]==N$ $\text{argv}[3]==d$ として与えられるとする($c < N$ を仮定して構わない)
- このときに平文 m を計算し、標準出力に十進数として書き出せ
(m の十進数表現と行末の改行 ¥ n 以外何も出力しないこと)
- 提出物: **ex14_5.c**

Ex14-6

～ファイルのRSA暗号化～

- ファイル名、RSA暗号の公開鍵 $N(=p*q)$, e が `argv[1]` `argv[2]` `argv[3]` にその順番で与えられるとする
- このとき、ファイルの内容を非負整数に変換しRSA暗号の平文 m とし、公開鍵 (N, e) を用いて暗号文 c を計算し、標準出力の第 1 行目に m を、第 2 行目に c を十進数として書き出せ
- ファイルから平文 m への変換を、課題14-1への自分の提出物と同じやり方で行うこと
- 変換して得られた m が N 以上で暗号化できない場合は第 1 行目の **m の値の後の第 2 行目に**「Too big.¥n」と表示せよ
- 提出物: **ex14_6.c**

Ex14-7（発展的課題）

～huge_intの改善～

- 構造体huge_intとその四則演算の関数について、改善案がある人は、(a) 提案する改善案の詳細と、(b) 提案する改善案が元のhuge_intより優れている点について、項目(a)と(b)に分けて述べよ
- 実行速度や占有メモリ量など、量を測れる改善案を提案する場合は、どの程度の改善が得られる（と予想される）か、具体的な数量を記述すること
- 繰り返しになるがhuge_int型定義ならびに関連関数を課題13-1以降で変更して用いる場合、型と関数の名前を変えて使用し、変更内容を14-7提出物で明記すること。名前だけ変更して提出物に含めて使うことは禁止する。提出物: [ex14_7.pdf](#)
- 松本は改善案を持っていないため解答例も無い可能性があります

課題の関係

入力方法・出力方法・計算の一覧

	入力元 (形式)	huge_intの計算	出力先 (形式)
ex13-1	argv (十進数)	和を計算	stdout (十進数)
ex14-1	file (文字列?)	—	stdout (十進数)
ex14-2	argv (十進数)	—	file (文字列?)
ex14-4	argv (十進数)	暗号化	stdout (十進数)
ex14-5	argv (十進数)	復号	stdout (十進数)
ex14-6	file (文字列?)	暗号化	stdout (十進数)

課題の関係

後の課題で流用できるように、関数を作って使いまわそう
太枠のところで新しく関数を作ればよい（ex14-6は楽勝？）

	入力元（形式）	huge_intの計算	出力先（形式）
ex13_1	argv(十進数)	和を計算	stdout (十進数)
ex14_1	file (文字列?)	—	stdout (十進数)
ex14_2	argv (十進数)	—	file (文字列?)
ex14_4	argv (十進数)	暗号化	stdout (十進数)
ex14_5	argv (十進数)	復号	stdout (十進数)
ex14_6	file (文字列?)	暗号化	stdout (十進数)

RSAの鍵の例

- $p=2063$
- $q=2099$
- $e=1048583$
- $d=387969$
- T2Scholaにより大きい鍵をrsa-key-medium.txt および rsa-key.txtとして十進数で与えている
- 他の鍵が欲しい場合メールで依頼して欲しい

オフィスアワーなど (要検討)

- 8月4日(金曜日) 13:40～16:30まで、オフィスアワーを計算機室において開催予定である。質問が無い場合参加する必要は無い—(~~オンライン併設???~~、~~詳細は次ページ~~)—
- ~matsumoto.r.aa/bin/check-final.shは最終課題に対応している。check-final.sh プログラム.c 引数1 引数2 ...のように起動すると引数1以降はargv[1] 以降になる

オンラインオフィスアワー

オフィスアワーに参加したいが、登校を禁じられた状態（かつ体調は良い）になった場合、東工大スラックを用いて連絡をすること

提出ファイルの一覧

7月28日（金） 13：30 〆切：

- ex13_1.c
- ex13_2.c

8月5日（土） 21：00 〆切：

- ex14_1.c
- ex14_2.c
- ex14_3.pdf
- ex14_4.c
- ex14_5.c
- ex14_6.c
- ex14_7.pdf（発展的課題）

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

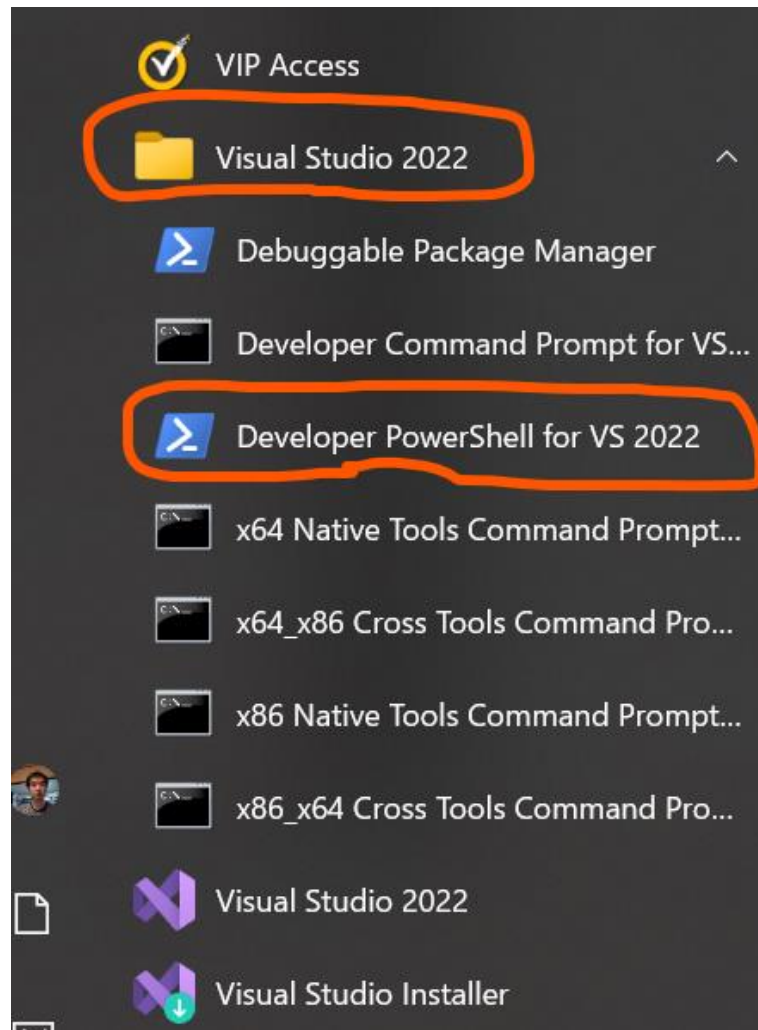
コロナ（等）対応のページ

登校不可能になったが体調は悪くなく、ウィンドウズで自宅実習する場合の情報です。体調不良で作業困難な場合は公欠制度を適用できる可能性があるので、**スラック**で教員に連絡して下さい。自宅でMacを用いる場合は前ページまでの情報で十分なはずです。まず第1回「参考資料」フォルダに従いVisual Studio 2022でCプログラムのコンパイル・実行できるようにして下さい

質問対応範囲

- 登校不可能になった人を除き、質問は計算機室でのみ受け付けます
- 登校不可能になった人については、ZOOMオンラインで講義時間ならびにオフィスアワー時間で受け付けます
- ZOOM質問において質問対応する開発環境は **MacOSのターミナルとWindowsパワースhellに限ります**。それ以外の開発環境でも最終課題に取り組めますが、使い方は自分で調べ自己責任で使ってください

パワースhell起動



コンパイル・実行

```
Developer PowerShell for VS 2022

PS C:\Users\山下隆太郎\Desktop\プログラミング発展2021→2022更新中\Final_Homework_Files> cl .\sieve.c .\hugeint.c
Microsoft(R) C/C++ Optimizing Compiler Version 19.32.31332 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

sieve.c
hugeint.c
コードを生成中...
Microsoft (R) Incremental Linker Version 14.32.31332.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:sieve.exe
sieve.obj
hugeint.obj
PS C:\Users\山下隆太郎\Desktop\プログラミング発展2021→2022更新中\Final_Homework_Files> dir

ディレクトリ: C:\Users\山下隆太郎\Desktop\プログラミング発展2021→2022更新中\Final_Homework_Files

Mode                LastWriteTime         Length Name
----                -
-a----          2022/07/05   22:41           5706 hugeint.c
-a----          2022/07/05   22:41           1883 hugeint.h
-a----          2022/07/05   23:08           6171 hugeint.obj
-a----          2022/07/05   22:41            614 make-testfiles.c
-a----          2022/07/05   22:41            369 rsa-key-medium.txt
-a----          2022/07/05   22:41            902 rsa-key.txt
-a----          2022/07/05   22:41           2468 sieve.c
-a----          2022/07/05   23:08        123904 sieve.exe
-a----          2022/07/05   23:08           2960 sieve.obj
-a----          2022/07/05   22:41        128063 最終課題のVS2022コンパイル手順.png

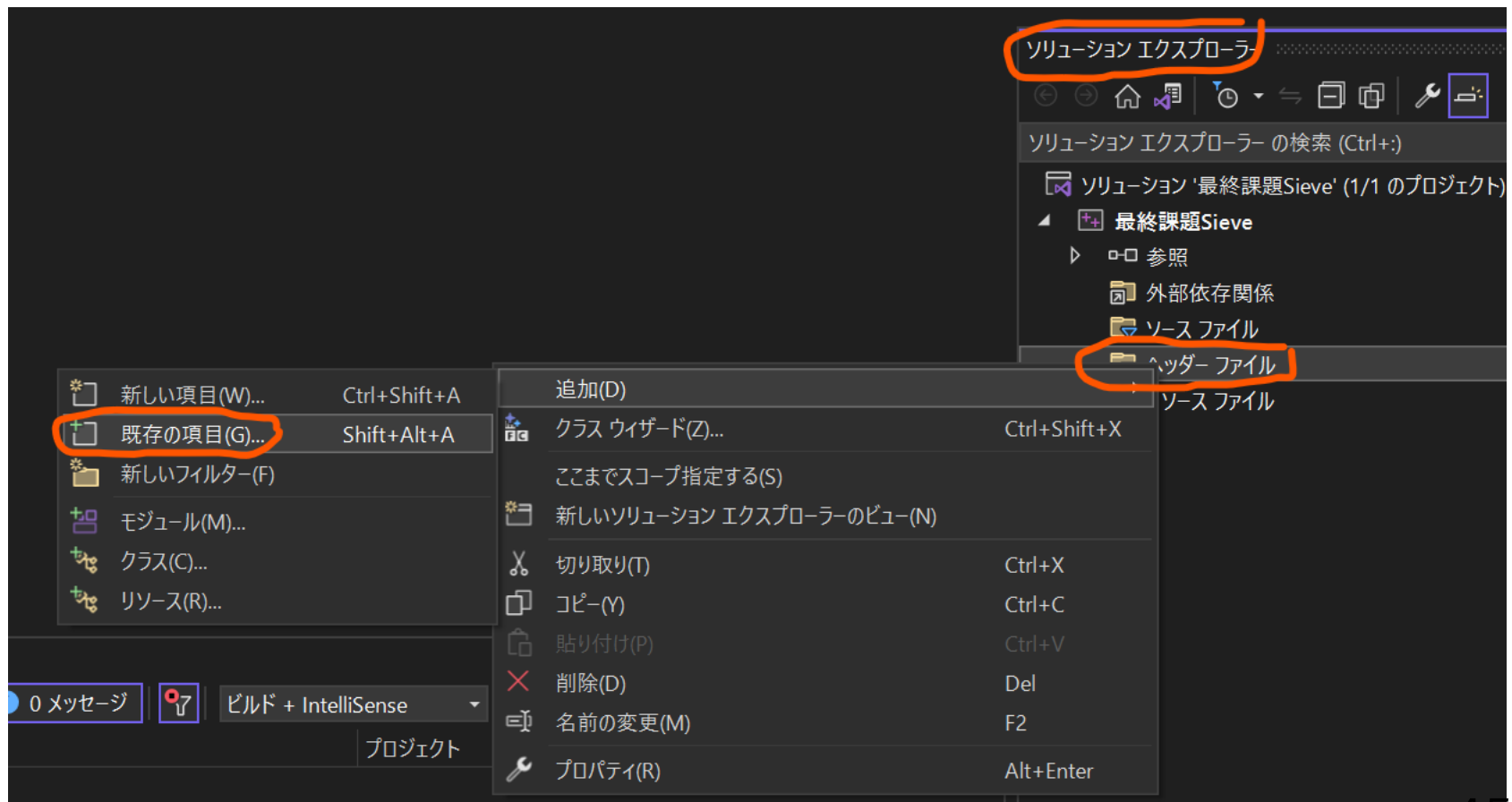
PS C:\Users\山下隆太郎\Desktop\プログラミング発展2021→2022更新中\Final_Homework_Files> .\sieve.exe 3 1
12460867
PS C:\Users\山下隆太郎\Desktop\プログラミング発展2021→2022更新中\Final_Homework_Files> _
```

コンパイル

実行

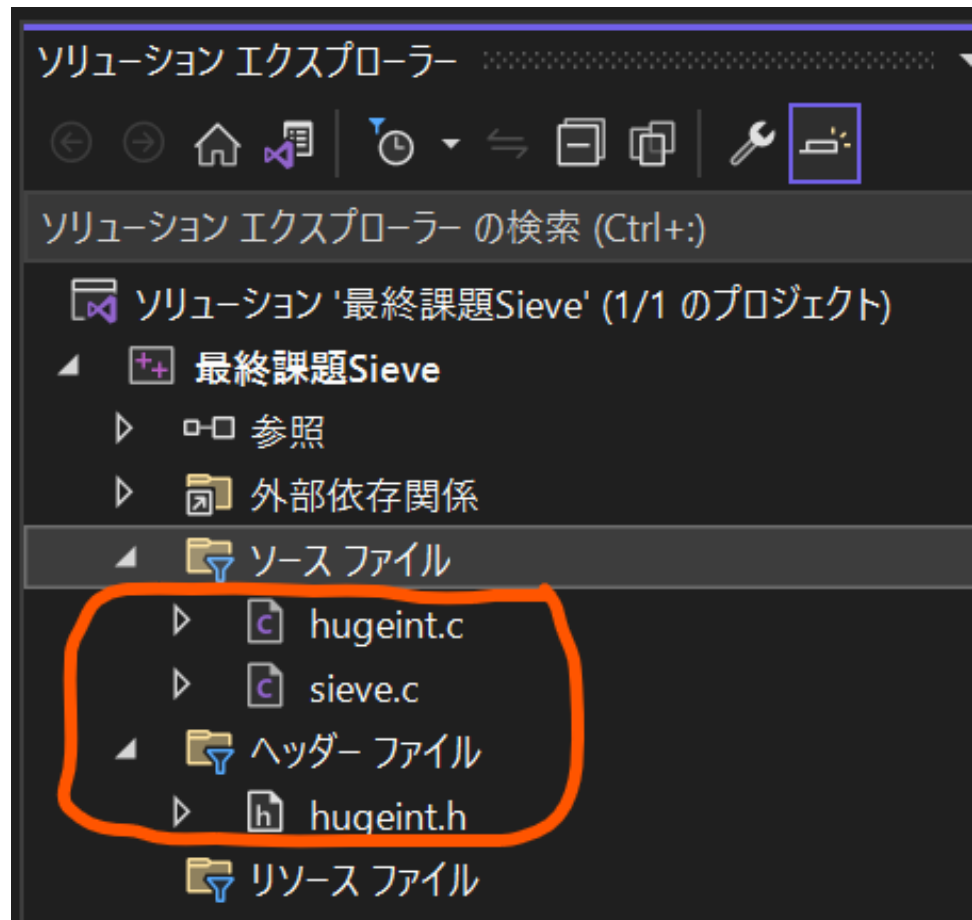
VSプロジェクトにhugeint.h追加

以降はVSの統合開発環境を用いる手順です
(質問は受け付けません)



VSプロジェクトに*.c追加

同様にhugeint.c とsieve.cを追加し以下の
ようにする



デバッグ無しで実行

必要なファイルを全部追加したら実行する



VSが結果を表示する



The screenshot shows the 'Microsoft Visual Studio デバッグ コンソール' (Debug Console) window. The console output consists of a list of prime numbers: 7681, 7687, 7691, 7699, 7703, 7717, 7723, 7727, 7741, 7753, 7757, 7759, 7789, 7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901, and 7907. Below this list, a message states: 'C:\Users\山下隆太郎\source\repos\最終課題Sieve\x64\Debug\最終課題Sieve.exe (プロセス 10628) は、コード 0 で終了しました。このウィンドウを閉じるには、任意のキーを押してください...' (C:\Users\山下隆太郎\source\repos\最終課題Sieve\x64\Debug\最終課題Sieve.exe (Process 10628) has terminated with code 0. Press any key to close this window...). The window has a standard Windows title bar with minimize, maximize, and close buttons.

```
Microsoft Visual Studio デバッグ コンソール
7681
7687
7691
7699
7703
7717
7723
7727
7741
7753
7757
7759
7789
7793
7817
7823
7829
7841
7853
7867
7873
7877
7879
7883
7901
7907
C:\Users\山下隆太郎\source\repos\最終課題Sieve\x64\Debug\最終課題Sieve.exe (プロセス 10628) は、コード 0 で終了しました。
このウィンドウを閉じるには、任意のキーを押してください...
```