

はじめてのC言語 練習帳

アトリエ未来 [著]



第三版

★1の易しい計算問題から

★5の双六、数当て、ポーカーまで全38問

C言語を愛する方々にお勧めの一冊です

はじめてのC言語 練習帳

[著] アトリエ未来

「技術書典 13 新刊」
令和四年九月廿三日 ver 3.0.0

■免責

本書は情報の提供のみを目的としています。

本書の内容を実行・適用・運用したことで何が起きようとも、それは実行・適用・運用した人自身の責任であり、著者や関係者はいかなる責任も負いません。

■商標

本書に登場するシステム名や製品名は、関係各社の商標または登録商標です。

また本書では、™、®、©などのマークは省略しています。

始めに

楽しいプログラミングの世界へようこそ。

情報技術「IT」に囲まれた生活を送るわたくしたち。多くの先人が築いた歴史の上に今日があります。^{こんにち}があります。

導入篇では、C言語が使われている場所や開発された方々、技術的な特徴などを紹介しています。

練習篇では、グラフの表示や日付などの計算、簡単なゲームの作成など様々な例を出題いたしましたので、楽しみながら、ご自身の腕試しとしてプログラミングに挑戦していただければと思います。

作成篇では、コーディングの参考例として著者のコードを示しました。

巻末には、今後の成長へ繋げて欲しいとの願いから、参考書籍とC言語の簡易まとめを付けました。^{つな}

現代の魔法、それがプログラミングです。自由自在にコンピュータを操って、幸せな未来へと大きく羽ばたいていってください。

♣ 対象読者

パソコンの操作ができる、初心者の方を想定しています。プログラミングに興味があって、なにか作ってみたい人の最初の一歩になればと願っています。^{*1}

C言語を学ぶ方の練習帳として、あるいは、誰かに教えたいと思っている方が学習希望者にお渡しするテキストとして、お読み頂ければ幸いです。

♣ プログラムのダウンロード

この本で作成したコードは、GitHub 上 ^{*2}で公開していますので、適宜ご利用下さい。

♣ 謝辞

Re:VIEW Starter^aを用いて、快適に執筆することができました。作者のkauplanさんに厚く御礼申し上げます。

^a <https://kauplan.org/reviewstarter/>



^{*1} コンピュータとして Mac をお使いの方を対象に執筆しております。Linux や Windows をお使いの方で、環境依存する箇所がございます場合には、適宜読み替えていただければ幸いです。

^{*2} https://github.com/Atelier-Mirai/c_source

また、表紙絵の女の子は、千葉県松戸市在住のフリーランス SD イラストレーター 早瀬ひろむ^aさんの作品です。素敵なイラストを描いてくださり、ありがとうございます。

^a <https://hiromu-hayase.tumblr.com>



早瀬ひろむ さん

背景の向日葵と海の絵は、がらくった^aさんの作品です。向日葵と青い空と青い海、とっても楽しくなります。

^a <https://www.ac-illust.com/main/profile.php?id=aromaru>



がらくった さん

また、信頼できる文献に触れて欲しいとの思いから、ウィキペディア等、各種文献より引用させていただいております。貢献に感謝するとともに、厚く御礼申し上げます。

♣ 著者紹介



卓越した技能を有する者として認められる国家資格「応用情報技術者」を保持。平成 30 年より「アトリエ未来^a」を創業。HTML 講座や Ruby 講座などプログラミングの個人指導や、IT パスポート講座等の資格講座の開催、ウェブサイト作成等を受注している。

趣味の将棋は、日本将棋連盟より三段の免状を允許。日本の美しい自然や豊かな精神性を宿す熊野古道を歩くことや、たくさんの花に囲まれた日々を愛している。

^a <https://atelier-mirai.net/>

【コラム】金の延棒クイズ

二進数の不思議を感じ、豊かな気分に成れるクイズです。(正解はこの本を最後まで読んでね。)



七日の給料の支払いとして金の延べ棒が一本あります。これを使って仕事をしてくれる方への日当を支払いたいと思います。

六回鋏を入れ七等分すれば日払いできますが、金の延べ棒を六回も切り取るのは大変です。

必要最小限の二回切り取ることをしたいですが、どことどこを切れば良いでしょうか？

目次

始めに

[コラム] 金の延棒クイズ	ii
-------------------------	----

第Ⅰ部 導入篇

1

第1章 C言語の紹介	3
1.1 暮らしに生きるC言語	3
1.2 C言語を創った人々	6
1.3 C言語の特徴	8
♣ 特徴	8
♣ 自由度	8
♣ 処理系の簡素化	9
♣ その他	10
♣ コード例	11
♣ 主な制御構造	11
♣ 誕生	11
♣ UNIX環境とC言語	11
♣ パソコンとC言語	11
♣ 現在のC言語	12
1.4 C言語の規格	13
♣ K&R	13
1.5 関連するプログラミング言語	14
♣ 先祖	14
♣ 繙承・拡張・サブセット	14
第Ⅱ部 練習篇	15
第2章 練習問題	17
2.1 難易度 ★☆☆☆☆	17
♣ いろいろな計算	17
♣ A4用紙の面積	17
♣ 「おはようございます」と挨拶	17
♣ 時刻に応じた挨拶	17
♣ 棒グラフの表示	17
♣ さいころ	18

♣ お天気予報	18
2.2 難易度 ★★☆☆☆	19
♣ 日々の積み重ね	19
♣ 定期預金	19
♣ 掛算九九	19
♣ 計算ゲーム	19
♣ 配列の要素の合計	19
♣ 配列の要素の最小値	19
♣ 配列の要素の並び替え	19
♣ 六十進数の計算	19
♣ 千支を求める	19
♣ 世界人口	19
♣ 二の幂乗	20
♣ 消費税	20
♣ 福引き	20
2.3 難易度 ★★★☆☆	21
♣ 合計が一万を越えるときの数	21
♣ 棒グラフ再び	21
♣ 千支を求める	21
♣ 円の面積	21
♣ 単語の長さ	21
♣ 計算ゲーム	22
♣ BMI	22
♣ 間年	22
♣ 漢数字	22
2.4 難易度 ★★★★☆	23
♣ 成績発表	23
♣ 素数	23
♣ 完全数	23
♣ 英単語帳アプリ	23
2.5 難易度 ★★★★★	24
♣ 生きてきた日数	24
♣ カレンダー	24
♣ 双六ゲーム	24
♣ 数当てゲーム (Match Number)	24
♣ ポーカー	24
第 III 部 作成篇	25
第 3 章 作成例	27

3.1	作成例 ★☆☆☆☆	27
	♣ いろいろな計算	27
	♣ A4 用紙の面積	27
	♣ 「おはようございます」と挨拶	28
	♣ 時刻に応じた挨拶	28
	♣ 棒グラフの表示	28
	♣ さいころ	30
	♣ お天気予報	33
3.2	作成例 ★★☆☆☆	35
	♣ 日々の積み重ね	35
	♣ 定期預金	35
	♣ 掛算九九	36
	♣ 計算ゲーム	36
	♣ 配列の要素の合計	38
	♣ 配列の要素の最小値	38
	♣ 配列の要素の並び替え	40
	♣ 六十進数の計算	42
	♣ 千支を求める	42
	♣ 世界人口	44
	♣ 二の幂乗	45
	♣ 消費税	46
	♣ 福引き	47
3.3	作成例 ★★★☆☆	49
	♣ 合計が一万を越えるときの数	49
	♣ 棒グラフ再び	50
	♣ 千支を求める	51
	♣ 円の面積	52
	♣ 単語の長さ	53
	♣ 計算ゲーム	54
	♣ BMI	63
	♣ 間年	64
	♣ 漢数字	67
3.4	作成例 ★★★★☆	71
	♣ 成績発表	71
	♣ 素数	72
	♣ 完全数	74
	♣ 英単語帳アプリ	75
3.5	作成例 ★★★★★	79
	♣ 生きてきた日数	79
	♣ カレンダー	83
	♣ 双六ゲーム	86

♣ 数当てゲーム (Match Number)	92
♣ ポーカー	94
付録 A 珠玉の名著のご紹介	109
♣ 教養としてのコンピューターサイエンス講義	109
♣ 達人プログラマー (第2版) 熟達に向けたあなたの旅	109
♣ コーディングを支える技術	110
♣ みんなのコンピューターサイエンス	110
♣ プログラマの数学	110
♣ C 言語による標準アルゴリズム事典	111
♣ アルゴリズム図鑑 絵で見てわかる 26 のアルゴリズム	111
♣ C の絵本—C 言語が好きになる 9 つの扉	111
♣ アルゴリズムの絵本-プログラミングが好きになる 9 つの扉	111
♣ 新・C 言語入門-シニア編-C 言語実用マスターシリーズ	112
♣ C 言語ポインタ完全制覇	112
♣ 小一時間でゲームをつくる 7 つの定番ゲームのプログラミングを体験	112
♣ リーダブルコード より良いコードを書くためのシンプルで実践的なテクニック	112
付録 B C言語 簡易まとめ	113
♣ コメント	113
♣ データ型	113
♣ リテラル	113
♣ 文字列	114
♣ 演算子	114
♣ 制御構造	116
♣ データアクセス	116
♣ 関数宣言	116
終わりに	119

第Ⅰ部

導入篇

第 1 章

C 言語の紹介

1.1 暮らしに生きるC言語

暮らしの中の様々なところで、コンピュータが使われています。そしてコンピュータプログラムの大きな部分を占めるのが「C言語」です。OS(オペレーティングシステム)や組み込み系など様々なところでC言語は使われています。



コンピュータ (Mac / iPad / iPhone)

なんといってもコンピュータの代表です。いろいろなウェブサイトを見たり、書類作成などお仕事に活用したり、そして「プログラミング」など。iPad でお絵描きや読書、iPhone で連絡を取り合ったり、写真や音楽、ゲームなどを楽しむことも出来ます。ロケットを打ち上げ、宇宙観測や天気予報に活かしたり、都市計画から住宅設計、工場で車や船、飛行機など様々なも

のを清算したり、音楽や映画、アニメーションなど芸術の分野に至るまで、様々な分野でコンピュータは使われています。

衣服

「天衣無縫」 - 「天人や天女の着物には縫い目がないという意から、詩文などが、よけいな修飾がなく、自然でわざとらしくなく完成されていること。また、人柄が純真で素直で、まったく嫌みがないさま。物事が完全無欠であることの形容 (goo 辞書)」

衣食住は、人が生きる上で生活の基盤となる要素です。大麻、木綿、絹糸、^{いにしえ}古の昔から日本人の身を纏う衣服に用いられてきました。縦糸と横糸を編んで一枚の布にして、布から切り取って縫いあわせて一着の衣服を仕立てていきます。コンピュータの活用により編み機から直接衣服を生み出すことが出来るようになりました。縫い目がないから着心地も良く、布の無駄もないとの特徴を持ちます。夢であった「天衣無縫」が顕現した瞬間です。

炊飯器

美味しい御飯を炊き上げてくれる炊飯器。「初めちよろちよろ、中ぱっぱ、赤子泣いてもふた取るな」と、朝早くから起きて竈で御飯を炊くのはなかなか難しいものでした。キャンプで飯盒炊飯した経験をお持ちの方もいらっしゃると思いますが、火加減が難しく焦げになってしまったり芯が残ってしまったこともあるかと思います。火力の調整をしたり、毎朝御飯が炊き上がるためのタイマー機能など、小さなコンピュータ（マイコン）を組み込むことで、美味しい御飯を頂けるようになりました。

エアコン

部屋にあるエアコン。暑い夏には涼風を、寒い冬には暖風を送り、快適に過ごせるよう室温を調整しています。部屋の温度を感じるセンサー機能、設定温度を保つように計算する演算機能、実際に風を送り出す送風機能から成り立っています。

給湯器

台所や浴室の給湯器もコンピュータの産物です。昔の人のお風呂として思い浮かぶのは、五右衛門風呂でしょうか。川や井戸から水を汲んできて、薪でお湯を沸かして、湯加減を確かめてと、お風呂は贅沢品でしたので、庶民は銭湯へ通いました。今ではとても簡単に給湯器を設定すると、浴槽一杯に給湯してくれ、温かいシャワーも使うことが出来ます。

信号機

会社や学校へ行く途中に見かける信号機。これにもコンピュータが使われています。赤信号、青信号と、色を変えるのはもちろん、道路の状況に応じて、近くの信号機と連携、青信号の長さを調整することで渋滞緩和を図るなどしています。

バス

通勤通学に電車やバスを利用する方も多いでしょう。例えば「さくら高校 行」と電光掲示板で行先表示したり、現金の他、ICカードを^{かざ}すことで乗車代金を払うことが出来ます。車両自体のガソリンや電気自動車の出力を制御する際や、バス停に「停留所を発車しました」と運行状況表示するなどの用途にも使われています。写真は、茨城県境町の「自動運転バス」^{*1}で、

^{*1} 自治体初！ 境町で自動運転バスを定常運行しています (<https://www.town.ibaraki-sakai.lg.jp/page/page00>)

地域の足として活躍しています。

こうして見てきただけでも、身の回りのいろいろなところにコンピュータが使われていることが分かります。他にはどこに使われているでしょうか。家の中で、お店で、学校や会社でなど、探してみましょう。

1.2

C言語を創った人々

C言語を開発した方々を Wikipedia から引用しつつ、ご紹介いたします。



ブライアン・カーニハン

ブライアン・カーニハンは、ベル研究所に在籍していたカナダ出身の計算機科学者である。C言語や UNIX の開発者であるデニス・リッチャー、ケン・トンプソンと共に、C言語および UNIX に対する多くの研究開発結果による貢献で知られている。

デニス・リッチャーと共に著の『プログラミング言語 C』(通称: K&R)は、事実上の規格書として扱われ、現在でも古典的な教科書の一つである。

多くのプログラミング言語入門書で、最初のプログラムとして書かれる `Hello world` は、彼がベル研究所で書いた B言語のチュートリアルで初めて使われた。



デニス・リッチャー

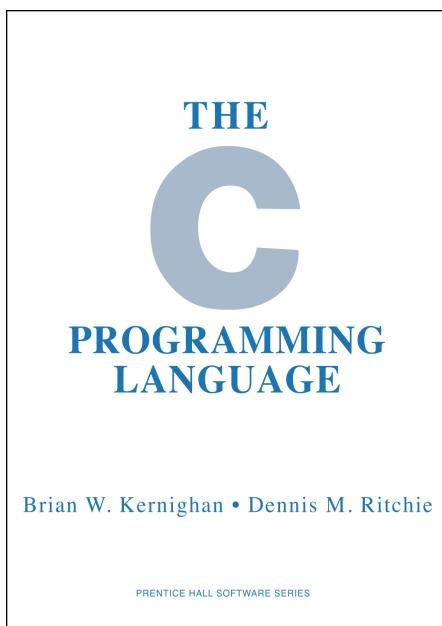
アメリカ合衆国の計算機科学者。1969年頃、同僚のケン・トンプソンと共に、ベル研究所で独自のオペレーティングシステム UNIX を作り始める。この UNIX 上で動作するアプリケーション作成の為に、トンプソンによって B言語が開発され、リッチャーがこれにデータ型と新しい文法等を追加し C言語が出来た。1973年にアセンブリ言語で書かれていた UNIX を C言語で書き換えることに成功した。C言語の開発は、リッチャーの UNIXへの最大の貢献である。

1983年、UNIX開発の功績により、ケン・トンプソンと共にチューリング賞を受賞している。今日、C言語は組込システムからスーパー・コンピュータまであらゆるプラットフォームで用いられ、彼の業績は偉大である。



ケン・トンプソン

ケン・トンプソンは、アメリカ合衆国の計算機科学者。長年ベル研究所に勤め、オリジナルの Unix を開発した。また C 言語の前身である B 言語を開発した。2006 年から Google で勤務しており、Go を共同開発した。他の主な業績として、正規表現、テキストエディタ QED と ed、UTF-8 コードの定義に加え、チェスの終盤定跡データベースやチェスマシン Belle の開発などコンピュータチェスへの貢献がある。1983 年に彼の長年の同僚であるデニス・リッチャーと共にチューリング賞を受賞した。「信用を信頼することについての考察」は、トンプソンハックとして知られる、セキュリティに関する重要な研究成果である。



▲図 1.1: プログラミング言語 C

1.3 C言語の特徴

C言語 シーゲンゴ は、1972年にAT&Tベル研究所のデニス・リッチー氏が主体となって開発したプログラミング言語です。

C言語の特徴などを Wikipedia から引用しつつ、ご紹介いたします。

♣ 特徴

- 汎用性が高い。プログラムの自由度や、目的に応じた拡張が容易であるため、パソコンソフトからゲームの作成、機械制御やシステム管理など、あらゆる分野に適応している。
- 対応する機器の範囲が広い。パソコンはもちろん、自動車や家電の組込み用マイコンからスーパーコンピュータまで、C言語を使用できるハードウェアは多様である。多目的性と、対応機器の多彩さのため、「コンピュータを使ってやること」は大抵、C言語で対応可能である。
- 商用・非商用を問わず、採用ソフトウェア分野が広い。作成や使用のための補助的なソフトウェアが豊富である。
- 開発時期が古いことから、文法に機械語の影響が強く、仕様自体は単純ではあるが明快ではなく難解である。この欠点を改良するためのうちに開発された後発言語に比較し、プログラマが記述しなければならないことが多く、低水準言語のように面倒で習得しにくい側面を持つ。
- アマチュアからプロ技術者まで、プログラマ人口が多く、プログラマのコミュニティが充実している。C言語は使用者の多さから、正負の両面含め、プログラミング文化に大きな影響を及ぼしている。
- C言語は手続き型言語である。コンパイラ言語とOSを念頭に設計している。ハードウェアをある程度抽象化しつつも、必要に応じて機械語やアセンブラーのコードと同じことを実現できるようなコンピュータ寄りの言語仕様になっている。低水準な記述ができる高級言語とも、高級言語の顔をした低級言語とも言うことがある。
- Cコンパイラは、移植の容易性、自由度、実行速度、コンパイル速度などを追求した。代わりにコンパイル後のコードの安全性を犠牲にしている。また、詳細を規格で規定せず処理系に委ねている部分が多く、C言語で書かれたソフトウェアでは処理系依存のコードが氾濫する原因となった。セキュリティー上の脆弱性や潜在的バグによる想定外の動作、コンパイラによる最適化の難しさといった問題を抱えており、最適化するとコンパイル速度が遅くなるなどの欠点が生じることがある。
- UNIXおよびCコンパイラの移植性を高めるために開発してきた経緯から、オペレーティングシステムカーネルおよびコンパイラ向けの低水準記述ができる。

♣ 自由度

- 文の区切りを終端記号セミコロン「;」で表し、改行文字にも空白にもトークンの区切りとしての意味しか持たせない「フリーフォーマット」という形式を採用している。中括弧{ }によ

るブロック構造およびスコープをサポートする。

- ALGOL の思想を受け継いで **構造化プログラミング** に対応している。手順を入れ子構造で示して見通しの良い記述をすることができる。
- モジュール化がファイルを単位として可能。モジュール内だけで有効な名前を使うことが出来るスコープを持っている。
- プログラムを戻り値つきのサブルーチンに分離できる。C言語ではこれを関数と呼び、関数内のプログラムコードでは、独立した変数が使用できる。これにより、データの流れがブロックごとに完結し、デバッグが容易になり、また関数の再帰呼び出しも可能となる。また、多人数での共同開発の際にも変数名の衝突が回避しやすくなる。
- C言語では、main関数と、標準ライブラリの printf, scanf 関数（およびその類型の関数）は、引数が可変という特殊な性格の関数である。K&Rでは、この特殊な関数 main と printf を使った例を最初に示している。
- システム記述言語として開発されたため、高級言語であるがアセンブラー的な低水準の操作ができる。ポインタ演算、ビットごとの論理演算、シフト演算などの機能を持ち、ハードウェアに密着した処理を効率よく記述できる。これはオペレーティングシステムやデバイスドライバなどを記述する上では便利であるが、注意深く利用しないと発見しにくいバグの原因となる。ライブラリ関数は、C言語規格が規定している関数と、OSが規定している関数との間の整合性、棲み分けなどが流動的である。
- 組み込みの整数型および浮動小数点数型のほか、構造体、共用体、列挙体（列挙型）によるユーザー定義のデータ型や列挙定数をサポートする。構造体および共用体はビットフィールドをサポートする。
- 多くの処理系がインラインアセンブラーを搭載しているほか、アセンブラーで出力したオブジェクトとのリンクが容易になっている。これにより速度が要求される部分だけをアセンブリ言語で記述するということが容易に行えることが多い。

♣ 処理系の簡素化

プログラムの内容によっては、以下に対して脆弱性対策を施しても実行速度の低下が無視できる程度であることも多く、欠点とみなされることも少なくない。

- 配列参照時の自動的な添字のチェックをしないこれを要因とする代表的なバグが、固定長のバッファ領域をはみだしてデータの書き込みが行われてしまう「バッファオーバーフロー」である。範囲外のアクセスは、書き込みだけでなく読み取りの場合も未定義動作を引き起こす。標準ライブラリにはバッファオーバーフローや範囲外アクセスを考慮していない関数があり、かつ多用されがちなため、しばしば脆弱性の原因となる。また、Cではプログラムにより明示的に制御（動的メモリ確保）することで可変長配列の実現を可能にしているが、確保した領域の範囲外にアクセスしても自動的な伸長は行なわれない。後継言語では、標準ライブラリまた

は組み込み型により可変長配列をサポートしていたり、範囲外アクセス時には例外（実行時エラー）を送出するなどして安全性を優先していたりすることが多い。

- 文字列を格納するための特別な型が存在しない文字列には `char` 型の配列を利用する。言語仕様上に特別な扱いはないが、ヌル文字 ('\0') を終端とする文字列表現を使い、その操作をする標準ライブラリ関数がある。これは実質的にメモリ領域のポインタアクセスそのもので、固定長バッファに対して、それより長い可変長の文字列を書き込んでしまうことがあり、バッファオーバーランの元凶の1つとなっている。後継言語では文字列処理を特に強化している場合が多く、標準ライブラリあるいは言語仕様による組み込みの文字列型を提供している。
- 自動変数の自動的な初期化をしない自動変数（静的でないローカル変数）は変数の中でも最も頻繁に用いられる。初期化されていない変数を参照した場合、値は不定となるが、不定な値へのアクセスは未定義の動作であるので、コンパイラ最適化の過程で想定しない形に改変することもある。変数宣言・初期化の仕様による制限から、変数宣言の時点では初期化せず後で代入することで初期化に代えることが日常的で、誤って不定の値の変数を読み出すバグを作り込みやすい。なお自動変数の自動とは変数の領域の確保と解放が自動であるという意味であり、自動的に初期化されるという意味ではない。後継言語では、明示的な初期化が記述されていない変数は、不定値ではなくその変数の型の既定値（ゼロあるいはゼロ相当の値）で初期化される仕様になっていることが多い。

♣ その他

- 文字の大文字・小文字を区別する。
- 入出力を含めほとんどの機能が、C言語自身で書かれたライブラリによって提供される。このことは、C言語の機種依存性が低く、入出力関係ライブラリをのぞいた部分は移植性（ポータビリティ）が高いことを意味する。さまざまな機種があるUNIXの世界でC言語が普及した理由のひとつである。
- プログラムの実行に必要とするハードウェア資源が、アセンブラーよりは多いが他の高級言語よりも少なくてすむため、現在さまざまな電化製品などの組み込みシステムでも使用されている。
- 組込み向けの場合は、プログラミング言語として、アセンブラー以外では、CとC++しか用意していないことがある。その場合、他のプログラミング言語は、CやC++で書かれた処理系が存在すれば、コンパイルすることにより利用可能となることもあるが、メモリ制約などで動作しないことがある。
- ANSI/ISOにより規格が標準化された後は言語仕様の変化が小さく安定していること、C言語のプログラマ人口やコード資産が多いこと、C++やObjective-CからC言語関数を直接利用できること、また必要に応じて他のプログラミング言語からC言語関数を呼び出すためのバインディングを記述することが容易であることなどから、APIの外部仕様としてC言語の関数インターフェイスが選ばれることが多い。例えばOpenGLやOpenCLのようなオープン規格は第一級言語としてC言語を採用している。

♣ コード例

▼ Hello world プログラム

```
#include <stdio.h>

int main(int argc, char* argv[]){
    printf("Hello, world!\n");
    return 0;
}
```

C 言語は `main` 関数から実行され、`printf` 関数は変数や書式化された文字列などが表示できる比較的高機能な出力関数である。コード中の「\n」は改行を表している。

♣ 主な制御構造

- `while` 文
- `do-while` 文
- `for` 文
- `if` 文
- `switch` 文
- 関数
- `return` 文

♣ 誕生

C 言語は、AT&T ベル研究所のケン・トンプソンが開発した B 言語の改良として誕生した。

1973 年、トンプソンと UNIX の開発を行っていたデニス・リッチャーは B 言語を改良し、実行可能な機械語を直接生成する C 言語のコンパイラを開発した。UNIX は大部分を C 言語によって書き換え、C 言語のコンパイラ自体も移植性の高い実装の Portable C Compiler に置き換わったこともあり、UNIX 上のプログラムはその後に C 言語を広く利用するようになった。

♣ UNIX 環境と C 言語

アセンブラーとの親和性が高いために、ハードウェアに密着したコーディングがやりやすかったこと、言語仕様が小さいためコンパイラの開発が楽だったこと、小さな資源で動く実行プログラムを作りやすかったこと、UNIX 環境での実績があり、後述の K&R といった解説文書が存在していたことなど、さまざまな要因から C 言語は業務開発や情報処理研究での利用者を増やしていく。特にメーカー間でオペレーティングシステムや CPU などのアーキテクチャが違う UNIX 環境では再移植の必要性がしばしば生じて、プログラムを C 言語で書いてソースレベル互換を確保することが標準となった。

♣ パソコンと C 言語

1980 年代に普及し始めたパーソナルコンピュータは当初、8 ビット CPU で ROM-BASIC を搭

載していたものも多く、BASIC が普及していたが、80年代後半以降、16ビット CPU を採用しメモリも増えた（ROM-BASIC 非搭載の）パソコンが主流になりだすと、2万円前後の安価なコンピュータが存在したこともあり、ユーザーが急増した。8ビットや 8086 系のパソコンへの移植は、ポインタなどに制限や拡張を加えることで解決していた。

♣ 現在の C 言語

1990年代中盤以降は、最初に学ぶプログラミング言語としても主流となった。GUI 環境の普及とオブジェクト指向の普及により Java、Objective-C、C++、PHP、Visual Basic、などの言語の利用者も増加したため、広く利用されるプログラミング言語の数は増加傾向にある。現在でも Java, C#, C++ など C 言語の後続言語を含めて、C 言語は比較的移植性に優れた言語であり、業務用開発やフリーソフトウェア開発、C++ などの実装が困難な組み込みなどの小規模のシステムで、幅広く利用されている。

1.4

C 言語の規格

♣ K&R

リッチャーとカーニハンの共著である「**The C Programming Language**」1978年を出版。その後標準ができるまで実質的なC言語の標準として参照。C言語は発展可能な言語で、この本の記述も発展の可能性のある部分は厳密な記述をしておらず、曖昧な部分が存在していた。C言語が普及するとともに、互換性のない処理系が数多く誕生した。これはプログラミング言語でしばしば起こる現象であり、C言語固有の現象ではない。

その後、C89/C90, C99, C11, C17と仕様が整備された。

1.5 関連するプログラミング言語

♣ 先祖

ALGOL

ヨーロッパ生まれのアルゴリズム記述言語。Pascal や C 言語などに影響を与えたとされる。

BCPL

MULTICS で作成された高級言語。

B 言語

初期の UNIX で作成されたインタプリタ方式の高級言語。BCPL を元に作られ、C の原型となつた。

♣ 繙承・拡張・サブセット

C++

C 言語を拡張してオブジェクト指向化したもの。Simula の影響を強く受けている。当初は C 言語のスーパー・セットだったが、現在は細かい部分において非互換仕様が増えている。

Objective-C

C 言語を拡張してオブジェクト指向化したもの。C 言語に Smalltalk のオブジェクトシステムを取り付けたような設計で、互換性は保たれている。C 言語からの拡張部分が C++ と干渉しないため、C++ と混在した記述が可能。

Java

C++ よりも言語文法レベルでオブジェクト指向を重視した言語。バッファオーバーランなどの危険性が高いポインタといったローレベルな要素を言語文法から排除している。仮想マシン (Java VM, JVM) 上で動作する。

C#

マイクロソフトが.NET Framework 向けに開発した言語。文法は C 言語および C++ に近い書式を持ち、Java と似ている部分も存在するが、機能的には Delphi がベースとなっている。

Rust

C 言語および C++ に代わるシステムプログラミング言語を目指している言語。言語レベルでの RAII の強制による自動メモリ管理機構を持ち、ガベージコレクション無しでも手動のメモリ管理が不要であり、実行性能は C/C++ と同等である。

Unified Parallel C

並列計算向けに C99 を拡張して作られた言語。

D 言語

C 言語をベースとし ABI 互換を保つつもり、テンプレートによるジェネリックプログラミングやオブジェクト指向プログラミング、関数型プログラミングなどをサポートするマルチパラダイムプログラミング言語である。

第 II 部

練習篇

第 2 章

練習問題

2.1 難易度 ★☆☆☆☆

♣ いろいろな計算

- $3 + 8 \times 3 - 1$ はいくつでしょうか？
- $18 \div 3 - 1$ はいくつでしょうか？
- $18 \div (3 - 1)$ はいくつでしょうか？
- 31を7で割った余りはいくつでしょうか？

♣ A4 用紙の面積

- 高さ 29.7 cm、幅 21.0 cm の四角形の面積は何 cm² でしょうか？

♣ 「おはようございます」と挨拶

「おはようございます」と挨拶するプログラムを作ってみましょう。

♣ 時刻に応じた挨拶

今の時刻を尋ねます。

朝なら「おはようございます」
昼なら「こんにちは」
夜なら「おやすみなさい」

と挨拶するプログラムを作ってみましょう。

♣ 棒グラフの表示

横棒グラフを表示してみましょう。

■ ■ ■ ■ ■	(5個 ■ 0個 □ があります)
■ ■ ■ ■ □	(4個 ■ 1個 □ があります)
■ ■ ■ □ □	(3個 ■ 2個 □ があります)
■ ■ □ □ □	(2個 ■ 3個 □ があります)
■ □ □ □ □	(1個 ■ 4個 □ があります)

いろいろな解き方がありますので考えてみましょう。

♣ さいころ

さいころを転がして、「偶数です」「奇数です」と表示するプログラムを作ってみましょう。

♣ お天気予報

傘を持っていった方がよいかどうか、アドバイスしてくれるプログラムを作りましょう。

降水確率20%なら 「傘はいらないです」
降水確率50%なら 「持って行った方がいいかも」
降水確率90%なら 「絶対持って行きましょう」

2.2 難易度 ★★☆☆☆

♣ 日々の積み重ね

毎日 1 %ずつこつこつ成長していったら、一年後には、何倍の実力になっているでしょうか？

♣ 定期預金

年利 3 %の定期預金に預けました。2 倍になるのは何年後でしょうか？年利 6 %なら、年利 12 %なら、2 倍になるのは何年後でしょうか？

♣ 掛算九九

掛算九九の表を出してみましょう。

♣ 計算ゲーム

計算ゲームです。10題 簡単な足し算（ $3 + 5 = ?$ など）が出題されるゲームを創りましょう。

♣ 配列の要素の合計

配列に 10 個の数字が入っています。合計を求めてみましょう。

♣ 配列の要素の最小値

配列に 10 個の数字が入っています。一番小さい数は何でしょうか？二番目に小さい数は何でしょうか？

♣ 配列の要素の並び替え

配列に 10 個の数字が入っています。大きい順から小さい順に並び替えてみましょう。

♣ 六十進数の計算

4680 秒は、何時間何分何秒でしょうか？

♣ 千支を求める

干支を求めてみましょう。2000 年生まれの人の干支は何でしょうか？

♣ 世界人口

世界の人口は 80 億人です。年に 1 %ずつ人口が増えると来年には何人になっているでしょうか？100 億人になる年はいつでしょうか？

♣ 二の幕乗

二の幕乗を出してみましょう。2の10乗はいくつでしょうか？ 2の20乗はいくつでしょうか？

♣ 消費税

税抜98円のアイスクリームを3個買うと消費税はいくら徴収されるでしょうか？ 日本人の年収の中央値は399万円です。消費税はいくら徴収されるでしょうか？

♣ 福引き

福引きアプリを創りましょう。

お客様>福引きをひかせてください。

店員>はい、どうぞ。

福引賞品：一等賞 世界一周の旅

：二等賞 温泉一泊二日

：三等賞 お好み焼き食べ放題

：残念賞 ティッシュペーパー

10回に1回は、世界一周の旅が当たるようにしてみましょう。各賞の当籤確率を変えるにはどのようにしたら良いでしょうか。

2.3 難易度 ★★★☆☆

♣ 合計が一万を越えるときの数

$1 + 2 + 3 + \dots$ と数を足していきましょう。どの数を足したときに、合計が 10000 を越えるでしょうか？

♣ 棒グラフ再び

縦棒グラフを出してみましょう。

□ □ ■ □ □ (ヒント)
□ ■ ■ □ □ 画面に表示するときは、
□ ■ ■ ■ □ 左から右、上から下が基本です。
□ ■ ■ ■ ■ 二次元配列を使って、
■ ■ ■ ■ ■ 表示する順番を工夫しましょう。

♣ 千支を求める

昭和 30 年生まれの方の干支は何でしょうか？ (S30 と入力します)

♣ 円の面積

円の面積を求めてみましょう。円周率は、3.141592653589 793238462643 383279502884... と続きます。よく使うので C 言語に用意されています。

```
#include <math.h> // 円周率 M_PI が定義されています
#include <stdio.h>

int main(){
    printf("円周率 = %f", M_PI); // 特に書式を指定しなかった場合
    printf("円周率 = %.4f", M_PI); // 小数点以下 4 衔表示
    printf("円周率 = %.15f", M_PI); // 小数点以下 15 衔表示
}
```

使ってみましょう。

♣ 単語の長さ

英単語の長さを求めるソフトです。入力された英単語の長さを求めてみましょう。

I -> 1文字
love -> 4文字
you -> 3文字

♣ 計算ゲーム

計算ゲームを創ってみましょう。二桁の数を出題したり、足し算の他、引き算や掛け算、割り算も出題されるようにしてみましょう。回答にかかった所要時間も表示させてみましょう。

♣ BMI

太りすぎ、痩せすぎの指標としてBMIがあります。体重 [単位 kg]/(身長 [単位 m] の二乗)で計算します。身長と体重を入れると、太りすぎか痩せすぎか分かるプログラムを作りましょう。基準は22が標準体重、25以上の場合は肥満、18.5未満である場合低体重です。

♣ 閏年

平年か閏年かを求めるプログラムを作ってみましょう。グレゴリオ暦が4で割り切れる年は閏年です。但し100で割り切れる年は閏年ではありません。しかしながら400で割り切れる年は閏年です。

♣ 漢数字

算用数字を漢数字にするプログラムを作ってみましょう。

```
302 -> 三百二  
122444000 -> 一億二千二百四十四万四千
```

2.4 難易度 ★★★★☆

♣ 成績発表

10人の名前が格納された配列と、10人の点数が格納された配列があります。成績の良い順に名前と点数を表示してみましょう。

♣ 素数

1と自分自身以外で割り切れない数のことを「素数」と言います。1～100までの間の素数を表示してみましょう。また1000個目の素数は何でしょうか？

♣ 完全数

完全数とは、自分自身の約数の和が自分自身になる数のことです。例を挙げます。

- 6は、1でも2でも3でも割り切れます。そして1と2と3を足すと6になります。
- 28は、1と2と4と7と14で割り切れます。1と2と4と7と14を足すと28になります。

28の次の完全数は、いくつでしょうか？10000までには、いくつ完全数があるでしょうか？

♣ 英単語帳アプリ

I	->	わたし
love	->	愛する
you	->	あなた

と答えられたら、満点です。10問出題して英語力向上を目指しましょう。

2.5 難易度 ★★★★☆

♣ 生きてきた日数

1980年1月1日生まれの人は、今日までに何日生きたことになるでしょうか？（生年月日は19800101と入力されます。）

昭和20年8月10日生まれの方は、どうでしょうか？（生年月日は3200810と入力されます。）

生後30000日目を迎えるのは、何年何月何日でしょうか？

♣ カレンダー

年と月を入力すると、その月のカレンダーを表示するプログラムを作ってみましょう。「ツェラーの公式」を用いると曜日を求めることが出来ます。（西暦1年1月1日は月曜日となりますので、7で割った余りを求めてることで、曜日が計算出来ます）

```
int y; // 西暦年
int m; // 月（但し1月、2月は前年の13月、14月として計算します）
int d; // 日
int h; // 曜日 // h が 0, 1, 2, 3, 4, 5, 6 の場合、
           // それぞれ 日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日
// 年月日をもとに曜日を算出する
h = (y + y/4 - y/100 + y/400 + (13*m+8)/5 + d) % 7;
```

♣ 双六ゲーム

双六ゲームを創ってみましょう。止まった升目に「3つ進む」や、「振り出しに戻る」も創ってみましょう。どこまで進んだか分かる表示機能や、オープニング・エンディングもあると楽しいですね。

♣ 数当てゲーム (Match Number)

事前に用意された3桁の数字を、ヒントをもとに当てていくゲームです。用意された数字が 925 だとします。999と入れると、9は正解ですので、1つ正解と表示します。520と入れると、5と2は正解ですので、2つ正解と表示します。592と入れると、（順番は違いますが）3つとも合っていますので、3つ正解と表示します。これを繰り返すことで、事前に用意された数字、925 を当てるゲームです。

♣ ポーカー

ポーカーを創ってみましょう。カードを配る機能、いらないカードを捨てる機能、他にはどんな機能が必要でしょうか？アスキーアートで、トランプの絵柄が表示されるともっと雰囲気が出ますね。

第 III 部

作成篇

第3章

作成例

練習課題の作成例です。いろいろな書き方で、プログラムを創ることが出来ます。コメントも入れてございますので、参考にしていただければ幸いです。

3.1 作成例 ★☆☆☆☆

♣ いろいろな計算

```
1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     // %d は整数型の書式指定子です。
5     // 計算結果を直接表示指せることが出来ます。
6     // 掛け算は「×」の代わりに「*」を使います。
7     printf("%d\n", 3 + 8 * 3 - 1);
8
9     // 変数に代入してから、出力することも出来ます。
10    int answer; // 変数の宣言。整数型の変数 answer を宣言しています。
11    answer = 18 / 3 - 1; // 変数への代入。
12                // 割り算は「÷」の代わりに「/」を使います。
13    printf("%d\n", answer);
14
15    int result = 18 / (3 - 1); // 直接、変数に初期値を代入することも出来ます。
16    printf("%d\n", result);
17
18    // 余りを求める演算（剰余演算子）として、「%」が用意されています。
19    printf("%d\n", 31 % 7);
20
21    return 0;
22 }
```

♣ A4 用紙の面積

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     double height = 29.7; // 小数を扱うには、double型として宣言します。
5     double width = 21.0;
6     double area;
7     area = height * width; // 掛け算は「×」の代わりに「*」を使います。
8     printf("%f\n", area); // 小数を表示する際は、%fを使います。
9
10    return 0;
11 }
```

♣ 「おはようございます」と挨拶

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     printf("おはようございます。\\n");
5     return 0;
6 }
```

♣ 時刻に応じた挨拶

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     // 変数の宣言
5     int what_time_is_it_now; // 今何時か、格納するための変数
6
7     printf("今何時ですか？\\n"); // 入力を促すために、メッセージを表示
8     scanf("%d", &what_time_is_it_now); // 整数型の数字を、受け取る
9
10    // 時刻に応じた挨拶をする
11    if (what_time_is_it_now < 12) { // 午前中
12        // 「\\n」は「改行文字」で、改行されます。
13        printf("おはようございます。\\n");
14    } else if (what_time_is_it_now < 18) { // 夕方まで
15        printf("こんにちは。\\n");
16    } else { // 夜なら
17        // puts関数を使うことも出来ます。
18        // 改行まで行ってくれます。
19        puts("おやすみなさい。");
20    }
21
22    return 0;
23 }
```

♣ 棒グラフの表示

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     // 横棒グラフを表示します。
5     // 直接、表示することも出来ます。
6     printf("■■■■■\n");
7     printf("■■■■□\n");
8     printf("■■■□□\n");
9     printf("■■□□□\n");
10    printf("■□□□□\n");
11
12    return 0;
13 }
```

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     // 変数名は、i でも良いですが、
5     // 行の変数であることが分かるようにすると良いです。
6     int row;          // 行
7     int black_box;   // 黒い■の数
8     int white_box;  // 白い□の数
9
10    for (row = 0; row < 5; row++) {
11        black_box = 5 - row; // 黒い■の数
12        white_box = row;    // 白い□の数
13        // 初期値は設定済みなので、省略出来ます。
14        for (; black_box > 0; black_box--) {
15            printf("■"); // ■を表示します。
16        }
17        for (; white_box > 0; white_box--) {
18            printf("□"); // □を表示します。
19        }
20        printf("\n"); // 改行します。
21    }
22
23    return 0;
24 }
```

```

1 #include <stdio.h>
2
3 // 自作の関数を創って、解くことも出来ます。
4
5 // 関数のプロトタイプ(原型)宣言
6 // 機能：黒い■と白い□を表示する
7 // 引数：int black_box // 黒い■の数
8 //         int white_box // 白い□の数
9 // 戻値：なし
10 void black_white_box(int black_box, int white_box);
11
12 int main(int argc, char const *argv[]) {
13     // 変数名は、i でも良いですが、
```

```

14 // 行の変数であることが分かるようにすると良いです。
15 int row; // 行
16 int black_box; // 黒い■の数
17 int white_box; // 白い□の数
18
19 for (row = 0; row < 5; row++) {
20     black_box = 5 - row; // 黒い■の数
21     white_box = row; // 白い□の数
22
23     // 黒い■と白い□を表示する関数を呼び出し、表示を任せます。
24     black_white_box(black_box, white_box);
25 }
26
27 return 0;
28 }
29
30 // プロトタイプ宣言を先頭でしているので、
31 // 自作関数本体を、main関数の後に書くことが出来ます。
32 void black_white_box(int black_box, int white_box) {
33     // C言語では、while(0)が偽となり、while文が終了します。
34     // ですので、このようにも書くことが出来ます。
35     while (black_box--) {
36         printf("■"); // ■を表示します。
37     }
38     while (white_box--) {
39         printf("□"); // □を表示します。
40     }
41     printf("\n"); // 改行します。
42 }
```

♣ さいころ

```

1 #include <stdio.h>
2 #include <stdlib.h> // rand関数が定義されています。
3 #include <time.h> // time関数が定義されています。
4
5 int main(int argc, char const *argv[]) {
6     int dice;
7
8     // 実行した時刻によって、異なった乱数となるように、
9     // 亂数の種を播きます。
10    srand(time(NULL));
11
12    // 6で割った余りに、1を加えることで、
13    // 1～6までの乱数が得られます。
14    dice = rand() % 6 + 1;
15    printf("サイコロの目は、%d です。\n", dice);
16
17    // 2で割った余りが0なら、偶数、そうでないなら奇数です。
18    if (dice % 2 == 0) {
19        printf("偶数です\n");
20    } else {
```

```
21     printf("奇数です\n");
22 }
23
24 // 論理和を使って、このように書くことも出来ます。
25 if (dice == 2 || dice == 4 || dice == 6) {
26     printf("偶数です\n");
27 } else {
28     printf("奇数です\n");
29 }
30
31 // switch case 文を使って書くことも出来ます。
32 switch (dice) {
33     case 1:
34         printf("奇数です\n");
35         // このbreak文がないと、
36         // case 2: も続けて実行されますので、
37         // break文を書いています。
38         break;
39     case 2:
40         printf("偶数です\n");
41         break;
42     case 3:
43         printf("奇数です\n");
44         break;
45     case 4:
46         printf("偶数です\n");
47         break;
48     case 5:
49         printf("奇数です\n");
50         break;
51     case 6:
52         printf("偶数です\n");
53         break;
54     default:
55         printf("エラーです\n");
56         // default文のbreak;は不要ですが、
57         // 対称性の観点から付けています。
58         break;
59 }
60
61 // 意図的に、break文を書かず、
62 // case 5: まで、スルーさせて、
63 // 奇数で在る旨、表示させています。
64 switch (dice) {
65     case 1:
66     case 3:
67     case 5:
68         printf("奇数です\n");
69         break;
70     case 2:
71     case 4:
72     case 6:
73         printf("偶数です\n");
```

```

74     break;
75 }
76
77 return 0;
78 }
```

(自作関数化した例)

```

1 #include <stdio.h>
2 #include <stdlib.h> // rand関数が定義されています。
3 #include <time.h> // time関数が定義されています。
4
5 // 機能：0～max未満の乱数を返す
6 // 引数：int max 乱数の最大値
7 // 戻り値：0～max未満の乱数
8 int random_number(int max) {
9     srand(time(NULL));
10    return (rand() % max);
11 }
12
13 int main(int argc, char const *argv[]) {
14     // 亂数はよく使うので、関数化してみました。
15     // 同じプログラムを何度も書かずには済むので、便利です。
16     // 関数のプロトタイプ宣言をしていないので、
17     // mainの前に、random_number関数は書かれている必要があります。
18     int dice = random_number(6) + 1;
19     printf("サイコロの目は、%d です。\\n", dice);
20
21     // 2で割った余りが0なら、偶数、そうでないなら奇数です。
22     if (dice % 2 == 0) {
23         printf("偶数です\\n");
24     } else {
25         printf("奇数です\\n");
26     }
27
28     return 0;
29 }
```

(外部ヘッダーファイルにした例) ??

```

1 #include <stdio.h>
2 #include <stdlib.h> // rand関数が定義されています。
3 #include <time.h> // time関数が定義されています。
4
5 // 自作したrandom_number()関数が定義されています。
6 // 自作のヘッダーファイルを読み込む際は、
7 // "(ダブルクォーテーション)"で囲みます。
8 #include "random_number.h"
9
10 int main(int argc, char const *argv[]) {
11     // random_number.hを読み込んでいるので、
12     // すぐに、random_number関数を使えます。
```

```

13 int dice = random_number(6) + 1;
14 printf("サイコロの目は、%d です。\\n", dice);
15
16 // 2で割った余りが0なら、偶数、そうでないなら奇数です。
17 if (dice % 2 == 0) {
18     printf("偶数です\\n");
19 } else {
20     printf("奇数です\\n");
21 }
22
23 return 0;
24 }
```

♣ お天気予報

```

1 #include <stdio.h>
2 #include <stdlib.h> // rand関数が定義されています。
3 #include <time.h> // time関数が定義されています。
4
5 // 機能：0～max未満の乱数を返す
6 // 引数：int max 亂数の最大値
7 // 戻り値：0～max未満の乱数
8 int random_number(int max);
9
10 // 0～max未満の乱数を返す関数
11 int random_number(int max) {
12     srand(time(NULL));
13     return (rand() % max);
14 }
15
16 int main(int argc, char const *argv[]) {
17     // 亂数はよく使うので、関数化してみました。
18     // 同じプログラムを何度も書かずに済むので、便利です。
19
20     // 降水確率 0 - 100 % の範囲の乱数
21     // (補完機能が働くので、長い変数名でも良いですが、
22     // さすがに長すぎるかもしれません)
23     int precipitation_probability = random_number(101);
24
25     // printf文の書式の中で、「%」を表示させるには、「%%」と記述します。
26     printf("降水確率は、%d %% です。\\n", precipitation_probability);
27
28     // if else 文の条件式は、
29     // 大きい順、あるいは、小さい順にして、
30     // 順次、条件に合致させるようにするとよいです。
31     if (precipitation_probability <= 20) {
32         printf("傘はいらないです\\n");
33     } else if (precipitation_probability < 90) {
34         printf("持って行った方がいいかも\\n");
35     } else {
36         printf("絶対持って行きましょう\\n");
37     }
```

```
38     return 0;  
39 }  
40 }
```

```
1 #include <stdio.h>  
2 #include <stdlib.h> // atoi関数用  
3  
4 int main(int argc, char const *argv[]) {  
5  
6     // コマンドラインから、降水確率を渡すことも出来ます。  
7     if (argc == 1) {  
8         printf("【使い方】\n");  
9         printf("傘を持っていくべきか、助言するプログラムです。\n");  
10        printf("もし降水確率が、30% なら、\n");  
11        printf("%s 30\n", argv[0]); // argv[0] はプログラム自身の名前です。  
12        printf("と入力して下さい。\n");  
13  
14     exit(1); // プログラムを終了します。  
15 }  
16  
17 // atoi関数は、文字列を数値に変換する関数です。  
18 // argv[1]が 文字列 "30" なら、  
19 // precipitation_probability には、整数 30 が入ります。  
20 int precipitation_probability = atoi(argv[1]);  
21  
22 if (precipitation_probability <= 20) {  
23     printf("傘はいらないです\n");  
24 } else if (precipitation_probability < 90) {  
25     printf("持って行った方がいいかも\n");  
26 } else {  
27     printf("絶対持って行きましょう\n");  
28 }  
29  
30 return 0;  
31 }
```

3.2 作成例 ★★☆☆☆

♣ 日々の積み重ね

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     double capability = 1.0; // 能力
5     int i;
6
7     for (i = 0; i < 365; i++) {
8         capability *= 1.01;
9         // capability = capability * 1.01; と書くことも出来ます。
10        // 自分自身に何かの計算を行い、
11        // その結果を自分自身に代入することは、よく行われるので、
12        // 代入と計算を一緒に行える、演算子がC言語には用意されています。
13    }
14
15    printf("一年後の能力は、\n");
16    printf("%.15f\n", capability); // 小数点以下15桁表示
17    printf("です。.\n");
18    printf("\n");
19    printf("C言語 double型での計算結果\n");
20    printf("37.783434332887275\n");
21    printf("\n");
22    printf("正確な値は、有効桁数732桁で、\n");
23    printf("37.");
24        "783434332887158877616604796497605460271135491591002003303933893694442"
25        "952198593811935639436889138752947230257466652966950262937798745172333"
26        "015079222338624286146825416806152531443969194556942776517247940062958"
27        "202175604957806833320549618283760329920784474440748232823522848774776"
28        "663377098517634258918092249275355047751709109700563151616706856329170"
29        "679969143031119841436101987303610665032253735962900715320344772671094"
30        "746342243980747288537748044810805431513656284728377150860725544069515"
31        "704180309669461071550627216255083200959680558767329997739256425299018"
32        "230968108183790782834451122341391699026728718809670675868494941801800"
33        "148043695322254918714677072113955042157310524945401321699479843200827"
34        "1425308713028897301180251050440194336501\n");
35    printf("です。.\n");
36    // 小数の計算は誤差がつきものです。
37    // double型で計算しましたが、有効桁数 14 桁でした。
38
39    return 0;
40 }
```

♣ 定期預金

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     double fixed_deposit = 1.0; // 定期預金
```

```

5 int year = 0;           // 預けた年数
6
7 // 繰り返す回数が分からない場合は、while文を使います。
8 // 定期預金が2倍未満の間、繰り返します。
9 while (fixed_deposit < 2.0) {
10     fixed_deposit = fixed_deposit * 1.03;
11     // fixed_deposit *= 1.03; // と書くことも出来ます。
12     year++;
13 }
14
15 printf("今年預けた定期預金は %d 年後に %5.3f 倍になりました。\n",
16       // 長くなった場合、カンマの後で改行することも出来ます。
17       year, fixed_deposit);
18
19 return 0;
20 }
```

♣ 掛算九九

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     int multiplier; // 乗数 (掛ける数)
5     int multiplicand; // 被乗数 (掛けられる数)
6
7     // 掛け算九九の表を表示する
8     printf(" 一 二 三 四 五 六 七 八 九\n");
9     printf(" の の の の の の の\n");
10    printf(" 段 段 段 段 段 段 段 段\n");
11    for (multiplier = 1; multiplier <= 9; multiplier++) {
12        for (multiplicand = 1; multiplicand <= 9; multiplicand++) {
13            printf("%3d", multiplicand * multiplier);
14        }
15        printf("\n");
16    }
17
18    return 0;
19 }
```

♣ 計算ゲーム

```

1 #include <stdlib.h> // rand関数が定義されています。
2 #include <time.h>   // time関数が定義されています。
3
4 // 機能：0～max未満の乱数を返す
5 // 引数：int max 乱数の最大値
6 // 戻値：0～max未満の乱数
7 int random_number(int max);
8
9 // 0～max未満の乱数を返す関数
```

```

10 int random_number(int max) {
11     srand(time(NULL));
12     return (rand() % max);
13 }
```

```

1 #include "util.h" // 自作関数いろいろの定義
2 #include <stdio.h>
3
4 int main(int argc, char const *argv[]) {
5
6     int operand1;          // 第一被演算子
7     int operand2;          // 第二被演算子
8     int operation_result; // 演算結果
9     int your_answer;      // 回答
10    int correct_answer;   // 正答数
11    int i;
12
13    correct_answer = 0; // 10問中何問正解か数えるために、初期化します。
14    // 1回目、2回目と表示させたいので、
15    // for (i = 1; i <= 10; i++) と書いているところに着目して下さい。
16    for (i = 1; i <= 10; i++) {
17        // 出題処理
18        operand1 = random_number(10);
19        // 二つの0～9までの数を用意します。
20        // do while文で、異なる数になるまで、繰り返します。
21        do {
22            operand2 = random_number(10);
23        } while (operand2 == operand1);
24
25        operation_result = operand1 + operand2; // 演算結果
26        printf("足し算ゲーム %d回目", i);
27        printf("%d + %d = ?\n", operand1, operand2);
28
29        // 回答を受け取る
30        scanf("%d", &your_answer);
31        while (getchar() != '\n')
32            ; // キーバッファ読み飛ばす
33
34        // 正解発表と正答数のカウント
35        if (your_answer == operation_result) {
36            printf("正解です。\\n");
37            correct_answer++;
38        } else {
39            printf("正解は、%d です。\\n", operation_result);
40        }
41    }
42
43    // 総合結果発表
44    printf("10問中 %d 問 正解です。\\n", correct_answer);
45
46    return 0;
47 }
```

♣ 配列の要素の合計

```

1 #include "util.h" // 自作関数いろいろの定義
2 #include <stdio.h>
3
4 int main(int argc, char const *argv[]) {
5     // 10個の数字が入った配列 繼めて初期値を与えています。
6     // 配列は、array(アレイ)と言います。
7     int array[] = {3, 15, 22, 81, 41, 0, 83, 72, 50, 33};
8     int sum; // 合計
9     int i;
10
11    // 合計処理
12    sum = 0; // 初期化します。
13    for (i = 0; i < 10; i++) {
14        sum = sum + array[i];
15        // sum += array[i]; と書くことも出来ます。
16    }
17
18    // 結果発表
19    for (i = 0; i < 10; i++) {
20        printf("array[%d]: %2d\n", i, array[i]);
21        // sum += array[i]; と書くことも出来ます。
22    }
23    printf(" 合計は %d です。\n", sum);
24
25    return 0;
26 }
```

♣ 配列の要素の最小値

```

1 #include "util.h" // 自作関数いろいろの定義
2 #include <limits.h> // 整数の取り得る範囲が定義されています。
3 #include <stdio.h>
4
5 int main(int argc, char const *argv[]) {
6     // 10個の数字が入った配列 繼めて初期値を与えています。
7     // 配列は、array(アレイ)と言います。
8     // (簡単化のために、同じ数字はないものと仮定しています。)
9     int array[] = {3, 15, 22, 81, 41, 83, 72, 0, 50, 33};
10
11    int min;      // 最小値を格納する変数
12    min = INT_MAX; // min = 999; と書いてもいいのですが、
13                // ここでは、整数型の取り得る最大値で初期化しています。
14    int i;
15
16    // 最小値を求める処理
17    // 最初、minに一番大きい数を入れておきます。
18    // そして、配列内のそれぞれの要素と比較することで、
19    // 一番、小さい数が、minにセットされます。
20    for (i = 0; i < 10; i++) {
21        if (min > array[i]) {
```

```

22     min = array[i];
23 }
24 }
25 // 結果発表
26 for (i = 0; i < 10; i++) {
27     printf("array[%d]: %2d \n", i, array[i]);
28 }
29 printf("一番小さい数は、%d です。 \n", min);
30
31 // 二番目に小さい数を求めます。
32 // int array[] = { 3, 15, 22, 81, 41, 83, 72, 0, 50, 33 };
33 // から、一番小さい数 0 を除いた
34 // int array[] = { 3, 15, 22, 81, 41, 83, 72, 50, 33 };
35 // の中から、一番小さい数を調べたら、良いことになります。
36 // 0を取り除くには、どうしたらよいのでしょうか？
37 // 取り除くのではなく、後ろの要素を前に詰める！と考えます。
38 // 0は7番目の要素ですから、
39 // array[7]にarray[8]を代入して、
40 // array[8]にarray[9]を代入すれば、詰めたことになります。
41 // そうすると、
42 // int array[] = { 3, 15, 22, 81, 41, 83, 72, 50, 33 };
43 // という配列になります。
44
45 // それでは、0がarrayの何番目の要素であったかを求めてみましょう。
46 int index = 0; // 最小値 min が何番目の要素であるか
47 for (i = 0; i < 10; i++) {
48     if (array[i] == min) {
49         index = i;
50     }
51 }
52
53 // indexには、7番目と入っています。
54 // 後ろの要素を前の要素に詰めます。
55 // array[7]にarray[8]を代入して、
56 // array[8]にarray[9]を代入します。
57 for (i = index; i < 9; i++) {
58     array[i] = array[i + 1];
59 }
60
61 // 詰まっているかどうか、確認の為に出力してみましょう。
62 printf("詰めた結果\n");
63 for (i = 0; i < 10; i++) {
64     printf("array[%d]: %2d \n", i, array[i]);
65 }
66
67 // 二番目に小さい数を調べます。
68 min = INT_MAX;
69 // 10まで調べなくても、9までOKです。
70 for (i = 0; i < 9; i++) {
71     if (min > array[i]) {
72         min = array[i];
73     }
74 }
```

```

75 // 結果発表
76 printf("二番目に小さい数は、%d です。\\n", min);
77
78 return 0;
79 }

```

♣ 配列の要素の並び替え

```

1 #include "util.h"    // 自作関数いろいろの定義
2 #include <limits.h> // 整数の取り得る範囲が定義されています。
3 #include <stdio.h>
4
5 int main(int argc, char const *argv[]) {
6     // 10個の数字が入った並び替え前の配列
7     int array[] = {3, 15, 22, 81, 41, 83, 72, 0, 50, 33};
8     int min; // 最小値を格納する変数
9     int i;
10
11    // 最小値を求める処理
12    min = INT_MAX;
13    for (i = 0; i < 10; i++) { // (a)
14        if (min > array[i]) {
15            min = array[i];
16        }
17    }
18
19    // それでは、0 が array の何番目の要素であったかを求めてみましょう。
20    int index = 0;           // 最小値 min が何番目の要素であるか
21    for (i = 0; i < 10; i++) { // (b)
22        if (array[i] == min) {
23            index = i;
24        }
25    }
26
27    printf("(a)(b)それぞれ行った場合の結果表示\\n");
28    for (i = 0; i < 10; i++) {
29        printf("array[%d]: %2d \\n", i, array[i]);
30    }
31    printf("min: %d index: %d\\n", min, index);
32
33    // ここで、(a) と (b) 同時に出来るのではと思えて来ます。
34    // 一緒に書くと、次のようにになります。
35    // 最小値を求めると同時に、最小値 min が何番目の要素であるか求める
36    min = INT_MAX;
37    for (i = 0; i < 10; i++) {
38        if (min > array[i]) {
39            min = array[i]; // (a)
40            index = i;      // (b)
41        }
42    }
43
44    printf("(a)(b)纏めて行った場合の結果表示\\n");

```

```

45   for (i = 0; i < 10; i++) {
46     printf("array[%d]: %2d \n", i, array[i]);
47   }
48   printf("min: %d index: %d\n", min, index);
49
50   // index には、7番目と入っています。
51   // 後ろの要素を前の要素に詰めます。
52   // array[7] に array[8] を代入して、
53   // array[8] に array[9] を代入します。
54   for (i = index; i < 9; i++) {
55     array[i] = array[i + 1];
56   }
57
58   // 前回は、array[9] は特に処理せず、そのままでしたが、
59   // 今回は、array[9] に、今求めた、一番小さい数を入れます。
60   array[9] = min;
61
62   // 一番小さい数が、array の最後に来ているはずです。
63   // 確認の為に出力してみましょう。
64   printf("一番小さい数が、最後に来ていることの確認\n");
65   for (i = 0; i < 10; i++) {
66     printf("array[%d]: %2d \n", i, array[i]);
67   }
68
69   // これで、一番小さい数を、最後に持って行くことが出来ました。
70   // 次は、0番目～8番目で並び替え、
71   // その次は、0番目～7番目で並び替え、
72   // ということを順番に繰り返していくと、全部の並び替えが完了します。
73
74   // 完成版のプログラムです。
75   // 0番目～何番目までを並び替えるのか、
76   // 管理するためのfor文で外側をくるんでいます。
77   // この並び替えのアルゴリズムを、選択ソート と言います。
78
79   int sorted; // 今、何番目の要素まで並び替えが終了しているのか。
80   for (sorted = 9; sorted >= 1; sorted--) {
81     min = INT_MAX;
82     for (i = 0; i <= sorted; i++) {
83       if (min > array[i]) {
84         min = array[i]; // (a)
85         index = i;      // (b)
86     }
87   }
88   for (i = index; i < sorted; i++) {
89     array[i] = array[i + 1];
90   }
91   array[sorted] = min;
92 }
93
94 printf("\n");
95 printf("【選択ソート】並び替え結果\n");
96 for (i = 0; i < 10; i++) {
97   printf("array[%d]: %2d \n", i, array[i]);

```

```

98 }
99
100 printf("\n");
101 printf("選択ソートの他に、有名な並び替え方法としては、\n");
102 printf("クイックソート\n");
103 printf("バブルソート\n");
104 printf("マージソート\n");
105 printf("が、あるので、学習してみて下さい。.\n");
106
107 return 0;
108 }
```

♣ 六十進数の計算

```

1 #include <stdio.h>
2 #include <time.h> // 時刻に関する関数がいろいろ用意されています。(今回未使用)
3
4 int main(int argc, char const *argv[]) {
5     int remain_time = 4680;
6     int hour;
7     int minute;
8     int second;
9
10    hour = remain_time / 3600; // 3600秒で割った商が時間です。
11    remain_time = remain_time % 3600; // 3600秒で割った余りが残り時間です。
12    minute = remain_time / 60; // 60秒で割った商が分です。
13    remain_time = remain_time % 60; // 60秒で割った余りが残り時間です。
14    second = remain_time;
15
16    printf("4680秒は、%d 時間 %d 分 %d 秒 です。.\n", hour, minute, second);
17
18 // 時刻に関する関数を使って求めることも出来ます。
19 // 興味のある方は、挑戦されて下さい。
20
21 return 0;
22 }
```

♣ 千支を求める

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     int year = 2000;
5
6     // switch case 文で、余りに応じて、それぞれの干支を表示します。
7     switch (year % 12) {
8         case 0:
9             printf("申年\n");
10            break;
11        case 1:
12            printf("酉年\n");
```

```

13     break;
14 case 2:
15     printf("戌年\n");
16     break;
17 case 3:
18     printf("亥年\n");
19     break;
20 case 4:
21     printf("子年\n");
22     break;
23 case 5:
24     printf("丑年\n");
25     break;
26 case 6:
27     printf("寅年\n");
28     break;
29 case 7:
30     printf("卯年\n");
31     break;
32 case 8:
33     printf("辰年\n");
34     break;
35 case 9:
36     printf("巳年\n");
37     break;
38 case 10:
39     printf("午年\n");
40     break;
41 case 11:
42     printf("未年\n");
43     break;
44 }
45
46 // 配列の各要素に、干支を格納すると、より簡潔に記述できます。
47
48 // 12で割った余りを配列の添字するのがポイントです。
49 // "申"と一文字に見えますが、
50 // 文字コードが utf-8 の場合、3文字(バイト) E7 94 B3 です。
51 // char saru[4] = { 'E7', '94', 'B3', '\n' };
52 // そして、干支は12種類ありますから、
53 // 配列の配列(=二次元配列)にすれば、干支の配列になります。
54 char eto[][4] = {"申", "酉", "戌", "亥", "子", "丑",
55                  "寅", "卯", "辰", "巳", "午", "未"};
56 // 配列の一次元目の[4]は必要です。
57 // 4を書かずに、[]とだけ書くと、コンパイルエラーとなります。
58 // ポインタを使って次のように書くこともできます。
59 // char *eto[] = { "申", "酉", "戌", "亥", "子", "丑",
60                  "寅", "卯", "辰", "巳", "午", "未" };
61
62 int index;
63 index = year % 12;
64 printf("あなたの干支は、%s です。\n", eto[index]);
65

```

```

66 // 一行に纏めて書くことも出来ます。
67 printf("あなたの干支は、%s です。\\n", eto[year % 12]);
68
69 return 0;
70 }
```

♣ 世界人口

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main(int argc, char const *argv[]) {
5
6     long population = 8000000000; // 世界人口
7                         // 約21億以上の数は、long型を使います。
8     double growth_rate = 0.01; // 人口増加率 1%
9
10    // int, double, long
11    // それぞれの型が自動的に変換されていることに着目して下さい。
12    long next_year_population = population * (1 + growth_rate);
13
14    // long型の書式指定子は、%ld です。
15    printf("来年の人口は %ld 人です。\\n", next_year_population);
16
17    // 何年後に100億人になるか、求めてみましょう。
18    // 繰返し回数が不明な場合は、while文を使います。
19    int year = 0;
20    while (population < 1e10) { // 1e10 は、10の10乗(=100億)です。
21        population *= (1 + growth_rate);
22        year++;
23        printf("%d 年後の人口は %ld 人です。\\n", year, population);
24    }
25
26    // おまけ
27    // 3桁ごとにカンマ区切りで出してみましょう。
28    // 数としての人口を文字列に変換します。
29    char str_population[32];           // 世界人口が入った文字列
30    char str_population_reverse[32]; // 逆順になっている文字列（作業用）
31    char str_comma_population_reverse
32        [32]; // カンマを入れて逆順になっている文字列（作業用）
33    char str_comma_population[32]; // カンマを入れた文字列
34    sprintf(str_population, "%ld", next_year_population);
35    int digit = strlen(str_population);
36    printf("%d 行の数です。\\n", digit);
37
38    // 0123456789
39    // str_population      : 8000000000
40    // str_comma_population : 8,000,000,000
41
42    // カンマは、最後の桁から3つごとに入れていきます。
43    // 最後から数えて3つずつカンマを入れていくのは、
44    // 扱いにくいので、先頭から3つずつ入れていけば良いように、
```

```

45 // 逆順に並び替えます。
46 // 8000000000 -> 0000000008
47 int i;
48 for (i = 0; i < digit; i++) {
49     str_population_reverse[digit - i - 1] = str_population[i];
50 }
51 // 文字列の終わりが分かるように最後に'\0'を入れます。
52 str_population_reverse[digit] = '\0';
53 printf("逆順に並び替えて、%s となりました。\n", str_population_reverse);
54
55 // 3桁ごとにカンマを入れていきます。
56 int comma = 0; // 今までに入れたカンマの数
57 for (i = 0; i < digit; i++) {
58     str_comma_population_reverse[i + comma] = str_population_reverse[i];
59     // 0桁目、1桁目では何もしませんが、
60     // 2桁目のコピーが終わった後に、
61     // カンマを追加する処理を行います。
62     if (i % 3 == 2) {
63         comma++;
64         str_comma_population_reverse[i + comma] = ',';
65     }
66 }
67 // 文字列の終わりが分かるように最後に'\0'を入れます。
68 str_comma_population_reverse[digit + comma] = '\0';
69 printf("カンマを追加して、%s となりました。\n", str_comma_population_reverse);
70
71 // もう一度、並び替えます。
72 for (i = 0; i < digit + comma; i++) {
73     str_comma_population[digit + comma - i - 1] =
74         str_comma_population_reverse[i];
75 }
76 // 文字列の終わりが分かるように最後に'\0'を入れます。
77 str_comma_population[digit + comma] = '\0';
78 printf("もう一度並び替えて、%s となりました。\n", str_comma_population);
79
80 return 0;
81 }
```

♣ 二の幂乗

```

1 #include <math.h> // pow 関数
2 #include <stdio.h>
3
4 int main(int argc, char const *argv[]) {
5     int n;
6     long power = 1;
7
8     // 2 の幂乗(べきじょう)を求めます。
9     for (n = 1; n <= 20; n++) {
10         power *= 2;
11         printf("2 の %2d 乗は %8ld です。\n", n, power);
12     }
```

```

13
14 // 幂乗を求める関数も用意されています。
15 for (n = 1; n <= 20; n++) {
16     power = pow(2, n);
17     printf("2 の %2d 乗は %8ld です。\\n", n, power);
18 }
19
20 // 16進数で出力する際は、%x を指定します。
21 // ここでは、power が long 型なので %lx を指定します。
22 // 10進数と16進数との対応も参考までに出力してみます。
23 printf("乗数 16進数          10進数\\n");
24 for (n = 1; n <= 16; n++) {
25     power = pow(2, n);
26     printf("%2d %11lx %13ld\\n", n, power, power);
27 }
28 for (n = 20; n <= 40; n += 10) {
29     power = pow(2, n);
30     printf("%2d %11lx %13ld\\n", n, power, power);
31 }
32
33 return 0;
34 }
```

乗数	16進数	10進数
1	2	2
2	4	4
3	8	8
4	10	16
5	20	32
6	40	64
7	80	128
8	100	256
9	200	512
10	400	1024
11	800	2048
12	1000	4096
13	2000	8192
14	4000	16384
15	8000	32768
16	10000	65536
20	100000	1048576
30	40000000	1073741824
40	10000000000	1099511627776

2の10乗は、1k(キロ) 約1000

2の20乗は、1M(メガ) 約100万

2の30乗は、1G(ギガ) 約10億

2の40乗は、1T(テラ) 約1兆

♣ 消費税

```

1 #include <math.h> // floor, ceil関数
2 #include <stdio.h>
3
4 int main(int argc, char const *argv[]) {
5
6     // 消費税を求めます。
7     int icecream = 98; // アイスクリームの値段
8     // 消費税率
9     // 変数にしておくと、変更があった場合の修正が楽です。
10    double consumption_tax_rate = 0.08;
11    double consumption_tax; // 消費税額
12
13    consumption_tax = icecream * 3 * consumption_tax_rate;
14    // %-5d で左詰で5桁表示されます。
15    printf("お買い上げ額は、 %-5d 円です。\n", icecream * 3);
16    printf("消費税は、 %6.2f 円です。\n", consumption_tax);
17    // 切り捨て floorは床の意味です。
18    printf("端数を切り捨てると、%6.2f 円です。\n", floor(consumption_tax));
19    // 切り上げ ceilは天井の意味です。
20    printf("端数を切り上げると、%6.2f 円です。\n", ceil(consumption_tax));
21
22    return 0;
23 }
```

♣ 福引き

```

1 #include "util.h" // 自作のrandom_number関数
2 #include <stdio.h>
3 #include <string.h>
4
5 int main(int argc, char const *argv[]) {
6     // 福引きの等級名
7     // 配列の添字は0から始まるので、先頭に""を入れています。
8     // prize_rank[1] が "一等賞"だと分かりやすいです。
9
10    char *prize_rank[] = {"", "一等賞", "二等賞", "三等賞", "残念賞"};
11    // 福引きの賞品
12    char *prize_item[] = {"", "世界一周の旅", "温泉一泊二日",
13                          "お好み焼き食べ放題", "ティッシュペーパー"};
14    // メッセージ
15    char message[64];
16
17    // 亂数はよく使うので、流用しています。
18    int lottery = random_number(10) + 1; // 福引き 1-10の乱数が得られます。
19
20    int rank; // 何等賞か？
21    if (lottery == 1) {
22        rank = 1; // 一等賞
23    } else if (2 <= lottery && lottery <= 3) {
24        rank = 2; // 二等賞
25    } else if (4 <= lottery && lottery <= 6) {
26        rank = 3; // 三等賞
```

```
27 } else {
28     rank = 4; // 残念賞
29 }
30
31 // 小さい順に切り取っているので、
32 // else if の下限は省略出来ます。
33 if (lottery == 1) {
34     rank = 1; // 一等賞
35 } else if (lottery <= 3) {
36     rank = 2; // 二等賞
37 } else if (lottery <= 6) {
38     rank = 3; // 三等賞
39 } else {
40     rank = 4; // 残念賞
41 }
42
43 // ちなみに条件式を書けないので、switch case 文には出来ません。
44
45 // それぞれの等級に応じたメッセージを創っています。
46 // 練習のために、文字列操作の関数を使っています。
47 strcpy(message, "おめでとう！"); // コピー関数
48 strcpy(message, prize_rank[rank]); // コピー関数
49 strcat(message, ":"); // 連結関数
50 strcat(message, prize_item[rank]); // 連結関数
51 strcat(message, " が当たったよ。"); // 連結関数
52 printf("%s\n", message);
53
54 return 0;
55 }
```

3.3 作成例 ★★★☆☆

♣ 合計が一万を越えるときの数

```

1 #include <stdio.h>
2
3 int main(int argc, char const *argv[]) {
4     int n;
5     int total;
6
7     // まず足してから10000を越えるか、判定したいので、
8     // do while 文を使いました。
9     // 後判定ループとも言います。
10    // 条件式が、total <= 10000 と
11    // 10000以下の間は繰り返すようになっている点も、着目して下さい。
12    n = 1;
13    total = 0;
14    do {
15        total = total + n;
16        n++;
17        // 一行に纏めて書くことも出来ます。
18        // total += n++;
19    } while (total <= 10000);
20    printf("%d を足したら、10000を越えて %d になりました。\n", --n, total);
21
22    // while 文を使って書くことも出来ます。
23    // 合計を求める処理が重複していることに着目して下さい。
24    n = 1;
25    total = 0;
26    total = total + n;
27    n++;
28    while (total <= 10000) {
29        total = total + n;
30        n++;
31    }
32    printf("%d を足したら、10000を越えて %d になりました.\n", --n, total);
33
34    // 重複を避けるために、次のように書くことも出来ます。
35    // 条件式が逆転していることに着目して下さい。
36    n = 1;
37    total = 0;
38    while (1) {
39        total = total + n;
40        n++;
41        if (total > 10000) {
42            break;
43        }
44    }
45    printf("%d を足したら、10000を越えて %d になりました.\n", --n, total);
46
47    return 0;
48 }
```

♣ 棒グラフ再び

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main(int argc, char const *argv[]) {
5
6     // 縦棒グラフを表示する
7     // y
8     // ^
9     // | □□■□□
10    // | □■■□□
11    // | □■■■□
12    // | □■■■■
13    // | ■■■■■
14    // -----> x
15
16    // 表示させたい棒グラフの値
17    int value[] = {1, 4, 5, 3, 2};
18    // グラフ用の二次元配列
19    char graph[5][5][6]; // [6]は、"□", "■"が、6文字必要なため。
20    // char graph[5][5]; // '*' , 'o' の一文字で表示するなら、これで良い。
21
22    // graph[0]
23    // □ graph[0][4]
24    // □ graph[0][3]
25    // □ graph[0][2]
26    // □ graph[0][1]
27    // ■ graph[0][0]
28
29    // value の値に応じて、棒グラフの長さをセットする
30    int x, y;
31    for (x = 0; x < 5; x++) {
32        int v = value[x];
33        for (y = 0; y < 5; y++) {
34            if (y < v) {
35                // graph[x][y] = "■"; // こう書きたいけれど、書けないので
36                // graph[x][y] = '*'; // char 一文字なら、こう書ける
37                strcpy(graph[x][y], "■");
38            } else {
39                strcpy(graph[x][y], "□");
40                // graph[x][y] = 'o';
41            }
42        }
43    }
44
45    // グラフ表示
46    for (y = 4; y >= 0; y--) {
47        for (x = 0; x < 5; x++) {
48            printf("%s", graph[x][y]);
49            // printf("%c", graph[x][y]); // '*' , 'o'
50            // の一文字で表示するなら、これで良い。
51        }
52        printf("\n");

```

```

53 }
54
55     return 0;
56 }
```

♣ 千支を求める

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int main(int argc, char const *argv[]) {
6     // 入力を促すメッセージ表示
7     printf("何年生まれですか? (例: S30) >\n");
8
9     // 8文字までキーボードから入力を受け取れるよう、バッファを用意
10    char buffer[8];
11
12    // キーボードから一行読み込む
13    // scanfはいろいろ支障があるので、
14    // fgets関数で、bufferへ、sizeof(buffer)-1文字分(7文字分),
15    // stdin(=キーボード)から読み込む
16    if (fgets(buffer, sizeof(buffer), stdin) == NULL) {
17        // エラーメッセージ出力
18        printf("キーボードから読み込めませんでした。.\n");
19        exit(1);
20    }
21
22    // 改行文字が含まれているかどうか？
23    if (strchr(buffer, '\n') != NULL) {
24        // S30エンター のように4文字タイプされたときは、
25        // 改行文字(エンター)を文字列の終端記号に置換する
26        buffer[strlen(buffer) - 1] = '\0';
27    } else {
28        // buffer内に、改行文字が含まれていない場合 (=8文字以上続けてタイプされた場合)
29    }
30    // 最初の7文字は読み込まれているので、残りの入力ストリーム(キーバッファ)をクリアする
31    while (getchar() != '\n')
32        ;
33
34    // 読み込んでいるか、確認
35    // printf("キーボードからの入力は、\n%s です。.\n", buffer);
36
37    // 入力文字列の先頭が、M, T, S, H で始まっているか確認し、西暦年に変換する
38    int year;           // 西暦年
39    char era_initial; // 明治: 'M', 大正: 'T', 昭和: 'S', 平成: 'H'
40    char era_year[3]; // 元号で何年か
41
42    // S30 と入力されていた場合、
43    // S をera_initialにセットする
```

```

44 era_initial = buffer[0];
45 // 30 をera_yearにセットする
46 era_year[0] = buffer[1];
47 era_year[1] = buffer[2];
48 era_year[2] = '\0';
49
50 char *endptr; // 数値に変換出来なかった文字
51 // atoi関数は、文字列を数値に変換出来なかった場合にでも、0を返します。
52 // 本当に、「0」という文字列が、0という数値に変換されたのか、
53 // 区別が出来ないので、strtol関数を使います。
54 year = strtol(era_year, &endptr, 10); // 10進数として変換する
55
56 // 数値変換不可の文字列の長さを調べる。
57 if (strlen(endptr) != 0) {
58     // エラーメッセージ出力
59     printf("%s は、元号年として認識出来ませんでした。\n", endptr);
60     exit(1);
61 }
62
63 switch (era_initial) {
64 case 'M':
65     year += 1867;
66     break;
67 case 'T':
68     year += 1911;
69     break;
70 case 'S':
71     year += 1925;
72     break;
73 case 'H':
74     year += 1988;
75     break;
76 default:
77     // エラーメッセージ出力
78     printf("%c は、元号の文字として認識出来ませんでした。\n", era_initial);
79     exit(1);
80 }
81
82 // 干支の算出方法は、前回と同様
83 // 干支の配列（12で割り切れる年は申年）
84 char *eto[] = {"申", "酉", "戌", "亥", "子", "丑",
85                 "寅", "卯", "辰", "巳", "午", "未"};
86 printf("%c%s年(%d年)生まれのあなたの干支は、%s です。\n", era_initial,
87         era_year, year, eto[year % 12]);
88
89 return 0;
90 }
```

♣ 円の面積

```

1 #include <math.h> // 円周率 M_PI が定義されています
2 #include <stdio.h>
```

```

3 int main(int argc, char const *argv[]) {
4     int radius = 10; // 円の半径
5     double area = pow(radius, 2) * M_PI; // 円の面積
6         // pow関数で、2乗を求めています。
7
8     printf("円周率 = %f\n", M_PI); // 特に書式を指定しなかった場合
9     printf("円周率 = %.4f\n", M_PI); // 小数点以下4桁表示
10    printf("円周率 = %.15f\n", M_PI); // 小数点以下15桁表示
11    printf("半径 %d の円の面積は、%.15f です。.\n", radius, area);
12
13    return 0;
14 }

```

♣ 単語の長さ

```

1 ****
2 * 入力された英単語の長さを表示する
3 * ( http://nzlife.net/archives/9581 に長い英単語の豆知識があります )
4 ****
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 // #define で、TRUE という定数を1であると定義しています
10 // プログラム中によく使う定数は、このように定義しておくと、
11 // 意味が分かりやすくて、よいです。
12 #define TRUE 1
13
14 int main(int argc, char const *argv[]) {
15     char buffer[64]; // 英単語の読み込み用
16
17     while (TRUE) {
18         // 入力を促すメッセージの表示
19         printf("英単語を入力して下さい。(終了:bye)\n");
20
21         // ちなみにプログラム学習用なので、
22         // scanf("%s", buffer);
23         // の一行が簡単だったりします。
24
25         // キーボードから一行読み込む
26         // fgets関数で、bufferへ、sizeof(buffer)-1文字分(7文字分),
27         // stdin(=キーボード)から読み込む
28         if (fgets(buffer, sizeof(buffer), stdin) == NULL) {
29             // エラーメッセージ出力
30             printf("キーボードから読み込めませんでした。\n");
31             exit(1);
32         }
33
34         // 改行文字が含まれているかどうか？
35         if (strchr(buffer, '\n') != NULL) {
36             // S30エンター のように4文字タイプされたときは、

```

```

37     // 改行文字（エンター）を文字列の終端記号に置換する
38     buffer[strlen(buffer) - 1] = '\0';
39 } else {
40     // buffer内に、改行文字が含まれていない場合（=8文字以上続けてタイプされた場合）
41     // 最初の7文字は読み込まれているので、残りの入力ストリーム（キーバッファ）をクリアする
42     while (getchar() != '\n')
43     ;
44 }
45
46 // 無限ループとなっているので、
47 // プログラム終了のための文字列 "bye" と比較します。
48 // strcmp は 比較した文字が小さいとき(辞書順に並べたときに前にくる場合) -1
49 // を strcmp は 比較した文字が大きいとき(辞書順に並べたときに後にくる場合)
50 // 1 を返します。そして、C言語では、0以外は真と判断しますので、if
51 // ((strcmp(buffer, "bye"))){ と書いても同じですが、!= 0
52 // と明示されていると、分かりやすいかと思います。
53 if ((strcmp(buffer, "bye")) != 0) {
54     printf("入力された英単語は、%s ですね。\n", buffer);
55     printf("長さは、%lu 文字の単語ですね。\n", strlen(buffer));
56     printf("意味は・・・ う～ん分かりません。.\n");
57     printf("\n"); // 前の行に\nを二つ入れてもOKですが、
58                         // 画面表示と見た目をあわせて置いた方が分かりやすいです。
59 } else {
60     printf("また、使ってね。bye-bye\n");
61     break;
62 }
63
64 return 0;
65 }
66 }
```

♣ 計算ゲーム

```

1 ****
2 * 足し算ゲームとほぼ同様です。
3 * 桁数と、演算の種類をリクエストするようになっています。
4 * 簡単なものを創って、より高機能のものへと、拡張していくと良いです。
5 ****
6 #include "util.h" // 自作の乱数
7 #include <math.h>
8 #include <stdio.h>
9 #include <string.h>
10
11 int main(int argc, char const *argv[]) {
12     int operand1;           // 第一被演算子
13     int operand2;           // 第二被演算子
14     char operator;          // 演算子 ('+', '-', '*', '/')
15     int digit;              // 桁数
16     int operation_result; // 演算結果
17     int your_answer;        // 回答
```

```

18 int correct_answer; // 正答数
19 int i;
20
21 // 桁数をリクエスト
22 do {
23     // scanf簡単です。
24     // 文字を入力されたときは、無視します。
25     printf("何桁の数字で挑戦しますか？\n");
26     printf("一桁: 1 二桁: 2 三桁: 3 四桁: 4\n");
27     scanf("%d", &digit);
28     while (getchar() != '\n')
29         ; // キーバッファ読み飛ばす
30 } while (digit <= 0 || digit >= 5);
31
32 // 演算子をリクエスト
33 do {
34     printf("どの計算に挑戦しますか？\n");
35     // 足し算、引き算、掛け算、割り算のことです。
36     // 技術者なら、
37     // 加算、減算、乗算、除算という呼び方も知っておいて欲しいです。
38     printf("加算: '+' 減算: '-' 乗算: '*' 除算: '/'\n");
39     scanf("%c", &operator);
40     while (getchar() != '\n')
41         ; // キーバッファ読み飛ばす
42     // char型 一文字なので、簡単に比較出来ます。
43     // 全角で "+" や "-" なら文字列操作関数を使って下さい。
44 } while (operator != '+' && operator != '-' && operator != '*'
45       && operator != '/');
46
47 correct_answer = 0; // 10問中何問正解か数えるために、初期化します。
48 // 1回目、2回目と表示させたいので、
49 // for (i = 1; i <= 10; i++) と書いているところに着目して下さい。
50 for (i = 1; i <= 10; i++) {
51     // 出題処理
52     // 署名を求める関数を使って、必要な桁数に調整します。
53     operand1 = random_number(pow(10, digit));
54     // do while文で、異なる数になるまで、繰り返します。
55     do {
56         operand2 = random_number(pow(10, digit));
57         // 0の割り算は定義されていないので、条件式を変更します。
58     } while (operand2 == operand1 || (operator == '/' && operand2 == 0));
59
60     switch (operator) {
61     // switch case文には、文字型も使えます。
62     case '+':
63         operation_result = operand1 + operand2;
64         break;
65     case '-':
66         operation_result = operand1 - operand2;
67         break;
68     case '*':
69         operation_result = operand1 * operand2;
70         break;

```

```
71     case '/':
72         operation_result = operand1 / operand2;
73         break;
74     }
75
76     // ゲーム名を表示するための場合分けです。
77     // こう書いた方が素直です。
78     // switch(operator){
79     // case '+':
80     //     printf("足し算ゲーム %d 回目", i);
81     //     break;
82     // case '-':
83     //     printf("引き算ゲーム %d 回目", i);
84     //     break;
85     // case '*':
86     //     printf("掛け算ゲーム %d 回目", i);
87     //     break;
88     // case '/':
89     //     printf("割り算ゲーム %d 回目", i);
90     //     break;
91 }
92
93 // char型が0-127までの数字として扱われることに着目した書き方
94 // 連想配列（ハッシュ）のご紹介
95 //
96 // ASCIIコード表を見ると、
97 // '+' 43
98 // '-' 45
99 // '*' 42
100 // '/' 47
101 // なっていますので、
102 // 47個+1個の大きさの文字列の配列を用意します。
103 char game_name[47 + 1][20];
104 strcpy(game_name['+'], "足し算");
105 // 書いて、配列の初期値を設定します。
106 // strcpy(game_name[43], "足し算");
107 // と書いたのと同じです。
108 strcpy(game_name['-'], "引き算");
109 strcpy(game_name['*'], "掛け算");
110 strcpy(game_name['/'], "割り算");
111 // 実際に printf("%s", game_name['+']);と書くと「足し算」と表示されます。
112 // このことをわきまえると、
113 printf("%sゲーム %d 回目\n", game_name[operator], i);
114 // と書けます。
115 //
116 // 配列は、添字として、数字をとりましたが、
117 // 文字を添字として使えるようになると、分かりやすかったり、
118 // 便利だったりします。
119 // game_name["+"] = "足し算";
120 // の様なイメージです。
121 // 連想配列（ハッシュ）として、用意されている言語もあります。
122 // 他言語学習の際の参考になさって下さい。
123
```

```

124 printf("%d %c %d = ?\n", operand1, operator, operand2);
125
126 // 回答を受け取る
127 scanf("%d", &your_answer);
128 while (getchar() != '\n')
129     ; // キーバッファ読み飛ばす
130
131 // 正解発表と正答数のカウント
132 if (your_answer == operation_result) {
133     printf("正解です。\n");
134     correct_answer++;
135 } else {
136     printf("正解は、%d です。\n", operation_result);
137 }
138 }
139
140 // 総合結果発表
141 printf("10問中 %d 問 正解\n", correct_answer);
142
143 return 0;
144 }
```

(より均一な乱数が出るように改良)

```

1 /*
2  A C-program for MT19937, with initialization improved 2002/1/26.
3  Coded by Takuji Nishimura and Makoto Matsumoto.
4
5  Before using, initialize the state by using init_genrand(seed)
6  or init_by_array(init_key, key_length).
7
8  Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
9  All rights reserved.
10
11 Redistribution and use in source and binary forms, with or without
12 modification, are permitted provided that the following conditions
13 are met:
14
15  1. Redistributions of source code must retain the above copyright
16      notice, this list of conditions and the following disclaimer.
17
18  2. Redistributions in binary form must reproduce the above copyright
19      notice, this list of conditions and the following disclaimer in the
20      documentation and/or other materials provided with the distribution.
21
22  3. The names of its contributors may not be used to endorse or promote
23      products derived from this software without specific prior written
24      permission.
25
26 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
27 "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
28 LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
29 A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
```

```

30 OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
31 EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
32 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
33 PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
34 LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
35 NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
36 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
37
38
39 Any feedback is very welcome.
40 http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
41 email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)
42 */
43
44 /*
45 The original version of
46 http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/CODES/mt19937ar.c was
47 modified by Takahiro Omi as
48 - delete line 47 "#include<stdio.h>"
49 - delete line 174 int main(void){...}
50 - change N -> MT_N
51 - change N -> MT_N
52 - change the file name "mt19937ar.c" -> "MT.h"
53 */
54
55 /* Period parameters */
56 #define MT_N 624
57 #define MT_M 397
58 #define MATRIX_A 0x9908b0dfUL /* constant vector a */
59 #define UPPER_MASK 0x80000000UL /* most significant w-r bits */
60 #define LOWER_MASK 0x7fffffffUL /* least significant r bits */
61
62 static unsigned long mt[MT_N]; /* the array for the state vector */
63 static int mti = MT_N + 1; /* mti==MT_N+1 means mt[MT_N] is not initialized */
64
65 /* initializes mt[MT_N] with a seed */
66 void init_genrand(unsigned long s) {
67     mt[0] = s & 0xffffffffUL;
68     for (mti = 1; mti < MT_N; mti++) {
69         mt[mti] = (1812433253UL * (mt[mti - 1] ^ (mt[mti - 1] >> 30)) + mti);
70         /* See Knuth TAOCP Vol2. 3rd Ed. P.106 for multiplier. */
71         /* In the previous versions, MSBs of the seed affect */
72         /* only MSBs of the array mt[]. */
73         /* 2002/01/09 modified by Makoto Matsumoto */
74         mt[mti] &= 0xffffffffUL;
75         /* for >32 bit machines */
76     }
77 }
78
79 /* initialize by an array with array-length */
80 /* init_key is the array for initializing keys */
81 /* key_length is its length */
82 /* slight change for C++, 2004/2/26 */

```

```

83 void init_by_array(unsigned long init_key[], int key_length) {
84     int i, j, k;
85     init_genrand(19650218UL);
86     i = 1;
87     j = 0;
88     k = (MT_N > key_length ? MT_N : key_length);
89     for (; k; k--) {
90         mt[i] = (mt[i] ^ ((mt[i - 1] ^ (mt[i - 1] >> 30)) * 1664525UL)) +
91             init_key[j] + j; /* non linear */
92         mt[i] &= 0xffffffffUL; /* for WORDSIZE > 32 machines */
93         i++;
94         j++;
95         if (i >= MT_N) {
96             mt[0] = mt[MT_N - 1];
97             i = 1;
98         }
99         if (j >= key_length)
100            j = 0;
101    }
102    for (k = MT_N - 1; k; k--) {
103        mt[i] = (mt[i] ^ ((mt[i - 1] ^ (mt[i - 1] >> 30)) * 1566083941UL)) -
104            i; /* non linear */
105        mt[i] &= 0xffffffffUL; /* for WORDSIZE > 32 machines */
106        i++;
107        if (i >= MT_N) {
108            mt[0] = mt[MT_N - 1];
109            i = 1;
110        }
111    }
112
113    mt[0] = 0x80000000UL; /* MSB is 1; assuring non-zero initial array */
114 }
115
116 /* generates a random number on [0,0xffffffff]-interval */
117 unsigned long genrand_int32(void) {
118     unsigned long y;
119     static unsigned long mag01[2] = {0x0UL, MATRIX_A};
120     /* mag01[x] = x * MATRIX_A for x=0,1 */
121
122     if (mti >= MT_N) { /* generate N words at one time */
123         int kk;
124
125         if (mti == MT_N + 1) /* if init_genrand() has not been called, */
126             init_genrand(5489UL); /* a default initial seed is used */
127
128         for (kk = 0; kk < MT_N - MT_M; kk++) {
129             y = (mt[kk] & UPPER_MASK) | (mt[kk + 1] & LOWER_MASK);
130             mt[kk] = mt[kk + MT_M] ^ (y >> 1) ^ mag01[y & 0x1UL];
131         }
132         for (; kk < MT_N - 1; kk++) {
133             y = (mt[kk] & UPPER_MASK) | (mt[kk + 1] & LOWER_MASK);
134             mt[kk] = mt[kk + (MT_M - MT_N)] ^ (y >> 1) ^ mag01[y & 0x1UL];
135         }
136     }
137
138     return y;
139 }

```

```

136     y = (mt[MT_N - 1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
137     mt[MT_N - 1] = mt[MT_M - 1] ^ (y >> 1) ^ mag01[y & 0x1UL];
138
139     mti = 0;
140 }
141
142 y = mt[mti++];
143
144 /* Tempering */
145 y ^= (y >> 11);
146 y ^= (y << 7) & 0x9d2c5680UL;
147 y ^= (y << 15) & 0xefc60000UL;
148 y ^= (y >> 18);
149
150 return y;
151 }
152
153 /* generates a random number on [0,0x7fffffff]-interval */
154 long genrand_int31(void) { return (long)(genrand_int32() >> 1); }
155
156 /* generates a random number on [0,1]-real-interval */
157 double genrand_real1(void) {
158     return genrand_int32() * (1.0 / 4294967295.0);
159     /* divided by 2^32-1 */
160 }
161
162 /* generates a random number on [0,1)-real-interval */
163 double genrand_real2(void) {
164     return genrand_int32() * (1.0 / 4294967296.0);
165     /* divided by 2^32 */
166 }
167
168 /* generates a random number on (0,1)-real-interval */
169 double genrand_real3(void) {
170     return (((double)genrand_int32()) + 0.5) * (1.0 / 4294967296.0);
171     /* divided by 2^32 */
172 }
173
174 /* generates a random number on [0,1) with 53-bit resolution*/
175 double genrand_res53(void) {
176     unsigned long a = genrand_int32() >> 5, b = genrand_int32() >> 6;
177     return (a * 67108864.0 + b) * (1.0 / 9007199254740992.0);
178 }
179 /* These real versions are due to Isaku Wada, 2002/01/09 added */

```

```

1 ****
2 * バージョンアップ版です。
3 * 亂数に偏りがあるので、
4 * メルセンヌツイスター (Mersenne Twister) 法を用いています。
5 * http://www.sat.t.u-tokyo.ac.jp/~omi/random\_variables\_generation.html
6 * 綺麗な乱数が得られることに着目して下さい。
7 * また、ゲームらしく、10題解くのにかかった時間も表示しています。
8 ****

```

```

9 #include "mt.h" // Mersenne Twister法による乱数
10 // 以下の関数が用意されている。
11 // genrand_int32() // 符号なし32ビット長整数
12 // genrand_int31() // 符号なし31ビット長整数
13 // genrand_real1() // 一様実乱数[0,1] (32ビット精度)
14 // genrand_real2() // 一様実乱数[0,1] (32ビット精度)
15 // genrand_real3() // 一様実乱数(0,1) (32ビット精度)
16 // genrand_res53() // 一様実乱数[0,1] (53ビット精度)
17 // AからBの範囲の整数の乱数が欲しいときは
18 // genrand_int32()%(B-A+1)+A;
19 // のような関数を用いればよい。
20 #include <math.h>
21 #include <stdio.h>
22 #include <string.h>
23 #include <time.h>
24
25 int main(int argc, char const *argv[]) {
26     int operand1;           // 第一被演算子
27     int operand2;           // 第二被演算子
28     char operator;          // 演算子 ('+', '-', '*', '/')
29     int digit;              // 桁数
30     int operation_result; // 演算結果
31     int your_answer;        // 回答
32     int correct_answer;      // 正答数
33     time_t start_time;       // ゲーム開始時刻
34     time_t finish_time;      // ゲーム完了時刻
35     int i;
36
37     // オープニング
38     printf("*****\n");
39     printf("*\n");
40     printf("*      計算ゲームへようこそ\n");
41     printf("*\n");
42     printf("*****\n");
43     printf("\n");
44
45     // 桁数をリクエスト
46     do {
47         printf("何桁の数字で挑戦しますか？\n");
48         printf("一桁: 1    二桁: 2    三桁: 3    四桁: 4 \n");
49         scanf("%d", &digit);
50         while (getchar() != '\n')
51             ; // キーバッファ読み飛ばす
52     } while (digit <= 0 || digit >= 5);
53
54     // 演算子をリクエスト
55     do {
56         printf("どの計算に挑戦しますか？\n");
57         printf("加算: '+' 減算: '-' 乗算: '*' 除算: '/'\n");
58         scanf("%c", &operator);
59         while (getchar() != '\n')
60             ; // キーバッファ読み飛ばす
61     } while (operator != '+' && operator != '-' && operator != '*' && operator != '/'

```

```
62     '/');
63
64 // 出題処理
65 correct_answer = 0; // 正答数初期化
66 time(&start_time); // ゲーム開始時刻の保存
67 for (i = 1; i <= 10; i++) {
68     // digit桁の乱数を求める
69     // genrand_int32 は 32bit(0-約43億) の整数型の乱数を返すので、
70     // 10(または100, 1000, 10000)で割った余りが得たい乱数となる。
71     // (int) は、キャストと呼ばれる。
72     // pow関数の戻り値はdouble型なので、int型へ変換している。
73     if (operator != '/') {
74         operand1 = genrand_int32() % (int)pow(10, digit);
75     } else {
76         // 除算時、同じ桁同士で演算すると、ほぼ0となるので、調整。
77         operand1 = genrand_int32() % (int)pow(10, digit + 1);
78     }
79     do {
80         operand2 = genrand_int32() % (int)pow(10, digit);
81     } while (operator == '/' && operand2 == 0); // 0 の除算は定義されていない
82
83 // 正答を用意
84 switch (operator) {
85 case '+':
86     operation_result = operand1 + operand2;
87     break;
88 case '-':
89     operation_result = operand1 - operand2;
90     break;
91 case '*':
92     operation_result = operand1 * operand2;
93     break;
94 case '/':
95     operation_result = operand1 / operand2;
96     break;
97 }
98
99 // 何回目のゲームか、表示
100 switch (operator) {
101 case '+':
102     printf("足し算ゲーム %d 回目\n", i);
103     break;
104 case '-':
105     printf("引き算ゲーム %d 回目\n", i);
106     break;
107 case '*':
108     printf("掛け算ゲーム %d 回目\n", i);
109     break;
110 case '/':
111     printf("割り算ゲーム %d 回目\n", i);
112     break;
113 }
```

```

115 // 出題する
116 printf("%d %c %d = ?\n", operand1, operator, operand2);
117
118 // 回答を受け取る
119 scanf("%d", &your_answer);
120 while (getchar() != '\n')
121     ; // キーバッファ読み飛ばす
122
123 // 正解発表と正答数のカウント
124 if (your_answer == operation_result) {
125     printf("正解です。\n");
126     correct_answer++;
127 } else {
128     printf("正解は、%d です。\n", operation_result);
129 }
130
131 // ゲーム完了時刻の保存
132 time(&finish_time);
133
134 // 総合結果発表
135 printf("*****\n");
136 printf("*          結果発表\n");
137 printf("*          \n");
138 printf("*          10問中 %d 問 正解\n", correct_answer);
139 printf("*          %ld 秒 でクリア！\n", finish_time - start_time);
140 printf("*          \n");
141 printf("*          おめでとうございます！\n");
142 printf("*          \n");
143 printf("*****\n");
144 printf("\n");
145
146 return 0;
147 }

```

♣ BMI

```

1 /*****
2 * BMI を算出
3 *****/
4 #include <math.h>
5 #include <stdio.h>
6
7 int main(int argc, char const *argv[]) {
8     // 変数宣言
9     int height; // 身長
10    int weight; // 体重
11    double bmi; // BMI
12
13    // 入力処理
14    printf("身長(cm)を入力して下さい。.\n");
15    scanf("%d", &height);

```

```

16 while (getchar() != '\n')
17 ;
18
19 printf("体重(kg)を入力して下さい。\\n");
20 scanf("%d", &weight);
21 while (getchar() != '\n')
22 ;
23
24 // BMI算出
25 // int型のheightを、キャスト演算子(double)を使って、
26 // double型に変換しています。
27 // 優先順位を明確にするため、
28 // ((double)height) を丸括弧で囲ってから、
29 // 100 で割っている点に着目して下さい。
30 bmi = weight / pow(((double)height) / 100, 2);
31
32 // 結果表示
33 printf("BMI は %4.2f です。\\n", bmi);
34 if (bmi < 18.5) {
35     printf("痩せすぎです。\\n");
36 } else if (bmi < 25) {
37     printf("標準です。\\n");
38 } else {
39     printf("肥満です。\\n");
40 }
41
42 return 0;
43 }
```

♣ 閏年

```

1 ****
2 * 閏年を算出します。
3 ****
4 #include <math.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 // 分かりやすさのために、TRUE, FALSE という定数を定義します。
10 // 定数は全て大文字で書く慣習です。
11 #define TRUE 1
12 #define FALSE 0
13
14 // 関数のプロトタイプ宣言
15 // 西暦年を渡して、閏年なら、TRUEを返す関数
16 int leap_year1(int year);
17 int leap_year2(int year);
18
19 int main(int argc, char const *argv[]) {
20     // コマンドラインから、引数を渡すことも出来ます。
21     if (argc == 1) {
```

```

22 printf("【使い方】\n");
23 printf(
24     "閏年(leap year)か、平年(common year)か、算出するプログラムです。\\n");
25 printf("%s 2016\\n", argv[0]); // argv[0] はプログラム自身の名前です。
26 printf("の様に西暦年で入力して下さい。\\n");
27
28 exit(1); // プログラムを終了します。
29 }
30
31 int year; // 西暦年
32 char *endptr; // 数値に変換出来なかった文字
33 // atoi関数は、文字列を数値に変換出来なかった場合にでも、0を返します。
34 // 本当に、「0」という文字列が、0という数値に変換されたのか、
35 // 区別が出来ないので、strtol関数を使います。
36 year = strtol(argv[1], &endptr, 10); // 10進数として変換する
37
38 // 数値変換不可の文字列の長さを調べる。
39 if (strlen(endptr) != 0) {
40     // エラーメッセージ出力
41     printf("%s は、西暦年として認識出来ませんでした。\\n", argv[1]);
42     exit(1); // エラーコード 1 として、異常であることを伝えて終了します。
43 }
44
45 // 閏年かどうかはよく使うので、自作の関数を創って、判定することにします。
46 // leap_year1 : 長いif文の関数
47 if (leap_year1(year) == TRUE) {
48     printf("閏年です。\\n");
49 } else {
50     printf("平年です。\\n");
51 }
52
53 // leap_year2 : 整理したif文の関数
54 if (leap_year2(year) == TRUE) {
55     printf("閏年です。\\n");
56 } else {
57     printf("平年です。\\n");
58 }
59
60 return 0;
61 } // main()関数の最後です。
62
63 // 西暦年を渡して、閏年なら、TRUEを返す関数
64 int leap_year1(int year) {
65     // 4で割り切れる年は閏年です。
66     // 但し100で割り切れる年は閏年ではありません。
67     // しかしながら、400で割り切れる年は、閏年です。
68
69     // 素直にif文で書くと次のようにになります。
70     // (if文の中にif文を書くことも出来ます。)
71     if (year % 4 == 0) {
72         // 4で割り切れる年は閏年ですが、例外があるので、例外を書きます。
73         if (year % 100 == 0) {
74             // 但し100で割り切れる年は閏年ではありません。とありますが、

```

```

75 // さらにこの例外があるので、例外を書きます。
76 if (year % 400 == 0) {
77     // しかしながら、400で割り切れる年は、閏年です。とあるので、
78     // TRUEを返します。
79     return TRUE;
80 } else {
81     // 400で割り切れなかった年です。
82     return FALSE;
83 }
84 } else {
85     // 100で割り切れなかった年です。
86     return TRUE;
87 }
88 } else {
89     // 4で割り切れなかった年です。
90     return FALSE;
91 }
92 }

93 // 西暦年を渡して、閏年なら、TRUEを返す関数
94 int leap_year2(int year) {
95     // 4で割り切れる年は閏年です。
96     // 但し100で割り切れる年は閏年ではありません。
97     // しかしながら、400で割り切れる年は、閏年です。
98
99     // 素直にif文を書くとすごく長くなってしまいました。
100    // if else が複雑になっていて、合っているのか間違っているのか、
101    // 確認するのも大変です。
102
103    // 4で割り切れる.....(a)
104    // 100で割り切れる.....(b)
105    // 400で割り切れる.....(c)
106    // と条件が複合しているので、
107    // 閏年かどうか.....(x)
108    // 表にして整理すると、分かりやすいです。
109    // (真偽値表、カルノー図と言います)
110
111    // 割り切ることを TRUE(T)
112    // 割り切れないことを FALSE(F) として、表にしてみましょう。
113    // 8とおりの組み合わせが出来ます。
114    // - が入っているところは、あり得ない組み合わせのところです。
115    // 4で割り切れなければ、当然、100でも400でも割り切れませんものね。
116
117
118    // a b c x
119    // F - - F ... (1)
120    // F - - F ... (2)
121    // F - - F ... (3)
122    // F - - F ... (4)
123    // T F - T ... (5)
124    // T F - T ... (6)
125    // T T F F ... (7)
126    // T T T T ... (8)
127

```

```

128 // 表を見ると、(5) と (6) は同じですので、
129 // (5) または (8)の場合で、閏年になることが分かります。
130 // つまり、
131 // 4で割り切れて、100で割り切れない年
132 // または、
133 // 4で割り切れて、100で割り切れて、400で割り切れる年(=400で割り切れる年)
134 // の場合に閏年になることが分かります。
135
136 // よって次のif文でよいことが分かります。
137 if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
138     return TRUE;
139 } else {
140     return FALSE;
141 }
142 // 1600, 1700, 2000, 2004, 2099, 2100年を入れて、確認してみましょう。
143 // (どのテストケースで確認すべきか、考えるのも大切です)
144 }
```

♣ 漢数字

```

1 ****
2 * 算用数字 302 を 漢数字にします。
3 * 無量大数までの変換が出来ます。
4 * 参考: https://ja.wikipedia.org/wiki/命数法
5 *
6 * 正規表現については、
7 * https://ja.wikipedia.org/wiki/正規表現
8 * http://rubular.com 参照
9 ****
10 #include <regex.h> // 正規表現 regcomp, regex
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <string.h>
14
15 // 漢字変換に用いる定数
16 // main の外側に書かれているので、どの関数からでも参照可能
17 // const は 変更禁止の為の修飾子
18 const char *digit[] = {"〇", "一", "二", "三", "四",
19                         "五", "六", "七", "八", "九"};
20 const char *unit1[] = {"千", "百", "十", ""};
21 const char *unit2[] = {"",           "万",           "億",           "兆",           "京",
22                       "垓",           "秭",           "穰",           "溝",           "澗",
23                       "正",           "載",           "極",           "恒河沙", "阿僧祇",
24                       "那由他", "不可思議", "無量大数"};
25
26 // 4桁の漢数字への変換を行う関数
27 char *num2kan(char *str);
28
29 int main(int argc, char const *argv[]) {
30
31     // コマンドラインからの引数がなければ、使い方表示
32     if (argc == 1) {
```

```

33 printf("【使い方】\n");
34 printf(
35     "算用数字を漢数字にするプログラムです。無量大数まで変換出来ます。\n");
36 printf("%s 302\n", argv[0]);
37 printf("の様に入力して下さい。.\n");
38 exit(1);
39 }
40
41 // コマンドライン引数チェック
42 // '0'-'9' 以外の文字が含まれていたら、エラー表示し、終了する。
43 // (正規表現を用いたが、一桁ずつ読み込み、「0」-「9」か、チェックしても良い)
44
45 // 正規表現のC言語でのコーディングについては、
46 // http://atomic.jpn.ph/prog/lang/regex.html 参照
47
48 char *reg = "[0-9]+"; // 全て数字であるか判定するための正規表現
49 regex_t regst;
50 // 正規表現のコンパイル
51 if (regcomp(&regst, reg, REG_EXTENDED)) {
52     return 1;
53 }
54 // argv[1]が、正規表現にマッチするか、実行
55 if (regexec(&regst, argv[1], 0, NULL, 0)) { // マッチしない場合
56     printf("%s は 算用数字として認識出来ませんでした。\n", argv[1]);
57     exit(1);
58 }
59 // コンパイル結果の開放は必須
60 regfree(&regst);
61
62 // 最大取り扱い桁数 9999無量大数( $10^{68}$ )までの数
63 if (strlen(argv[1]) > 4 + 68 + 1) {
64     printf("%s は 長すぎます。73桁までの数字を入力して下さい。.\n", argv[1]);
65     exit(1);
66 }
67
68 // 漢数字への変換処理
69 // 4桁ごとに漢数字に変換する
70 // 例) 12 3456 7890
71 // => 十二 億 三千四百五十六 万 七千八百九拾
72
73 // 変数宣言
74 int digit_length; // 何桁の数字か
75 int i, u;
76 char string_number[4 + 68 + 1 + 1]; // 算用数字 最大取り扱い桁数
77 // 9999無量大数( $9999 \times 10^{68}$ )まで
78 strcpy(string_number, ""); // 初期化
79 // 変換後の漢数字(4桁毎に7文字21バイトになるので) また万億兆などの文字数も追加
80 char chinese_numeral[(7 * 3) * (72 / 4) + sizeof(unit2) + 1];
81 strcpy(chinese_numeral, "");
82 char work[7 * 3 + 1];
83 strcpy(work, "");
84
85 // 4桁ごとに区切って渡せるよう、先頭に0を付与する

```

```

86 // 12 3456 7890 => 0012 3456 7890
87 digit_length = strlen(argv[1]);           // 何桁の数であるか？
88 int giving_zero = 4 - (digit_length % 4); // 先頭に付与すべき0の数
89 if (giving_zero == 4) {
90     giving_zero = 0;
91 };
92 for (i = 0; i < giving_zero; i++) {
93     strcat(string_number, "0");
94 }
95 strcat(string_number, argv[1]);
96
97 // 何桁の数字か
98 digit_length = strlen(string_number);
99
100 // 4桁毎に漢数字に変換
101 // 4桁までなら unit2 "" // 8桁までなら unit2 "万" // 12桁までなら unit2 "億"
102 // を付与する
103 int unit2_index = (digit_length / 4) - 1;
104 for (u = 0; u < digit_length / 4; u++, unit2_index--) {
105     // string_number から 各ユニット毎に、4文字ずつ work にコピー
106     work[0] = string_number[4 * u + 0];
107     work[1] = string_number[4 * u + 1];
108     work[2] = string_number[4 * u + 2];
109     work[3] = string_number[4 * u + 3];
110     work[4] = '\n';
111
112     // 4桁毎に漢数字に変換、結果を連結する
113     strcat(chinese_numeral, num2kan(work));
114     // ... "兆", "億", "万", "" の付与
115     strcat(chinese_numeral, unit2[unit2_index]);
116 }
117
118 // 0 の場合のみ、空文字列のままなので、"○" にする。
119 if (strlen(chinese_numeral) == 0) {
120     strcpy(chinese_numeral, digit[0]);
121 }
122
123 // 結果表示
124 printf("%s\n", chinese_numeral);
125
126 return 0;
127 }
128
129 // 4桁 の 漢数字へ変換する関数
130 char *num2kan(char *str) {
131     // 変数宣言
132     int i;
133     char work[7 * 3 + 1]; // 三千四百五十六 7文字*3バイト+終端文字1バイト
134     strcpy(work, ""); // 初期化
135
136     // 一つずつ漢数字に変換する
137     for (i = 0; i < 4; i++) {
138         switch (str[i]) {

```

```
139     case '0':
140         break; // 何もせず、次の桁へ進む
141     case '1':
142         // 一の位のみ "一"と書く
143         if (i == 3) {
144             strcat(work, digit[str[i] - '0']); // str[i]-'0' 文字0から数値0へ変換
145         }
146         strcat(work, unit1[i]); // "千", "百", "十", "" のいずれか
147         break;
148     default:
149         strcat(work, digit[str[i] - '0']); // "一" ~ "九"
150         strcat(work, unit1[i]); // "千", "百", "十", "" のいずれか
151         break;
152     }
153 }
154
155 // 作業結果を戻す
156 return strcpy(str, work);
157 }
```

3.4 作成例 ★★★★☆

♣ 成績発表

```

1  ****
2  * 10人の名前が格納された配列と、10人の点数が格納された配列があります。
3  * 成績の良い順に名前と点数を表示してみましょう。
4  *
5  * C言語には、クイックソート(quicksort)と呼ばれるアルゴリズムで
6  * 実装された関数が標準で用意されていますので、今回はこれを用います。
7  * (前回、作成したプログラムを流用してももちろんOKです)
8  *
9  * クイックソートに関しては、
10 * https://ja.wikipedia.org/wiki/クイックソート
11 * を参照して下さい。
12 *
13 * qsortの利用方法については、
14 * http://www.cc.kyoto-su.ac.jp/~yamada/ap/qsort.html より、引用
15 ****
16 #include <stdio.h>
17 #include <stdlib.h> // quicksort
18 #include <string.h>
19
20 // 比較関数(大小判断を返す関数)
21 int compare_int(const void *a, const void *b) {
22     // b > a なら 正の数を返す。(降順)
23     return *(int *)b - *(int *)a;
24 }
25
26 int main(int argc, char const *argv[]) {
27     // 変数宣言
28     char *name[] = {"亜希子", "加世子", "小夜子", "妙子", "奈美子",
29                     "太郎", "次郎", "三郎", "四郎", "五郎"};
30     int score[] = {99, 88, 77, 66, 55, 55, 64, 73, 82, 91};
31     int backup[10];
32     int i, j, s;
33
34     // 結果表示
35     printf("並び替え前:\n");
36     for (i = 0; i < 10; i++) {
37         printf("%d %s\n", score[i], name[i]);
38     }
39
40     // 並び替えすると、元の配列が破壊されるため、バックアップを取る
41     // for文で一要素ずつコピーしても良いが、
42     // memcpy関数を紹介
43     memcpy(backup, score, sizeof(int) * 10); // int型10要素分をbackupへコピー
44
45     // 確認用
46     // printf("score backup\n");
47     // for (i = 0; i < 10; i++){
48     //     printf("%d %d\n", score[i], backup[i]);

```

```

49 // }
50
51 // 並び替えを行う
52 // 並び替えたい配列名、要素数、配列1要素分のサイズ、大小比較に用いる関数名
53 qsort(score, 10, sizeof(int), compare_int);
54
55 // 確認用
56 // printf("並び替え後:\n");
57 // for (i = 0; i < 10; i++){
58 //   printf("%d\n", score[i]);
59 // }
60
61 printf("並び替え後:\n");
62 // 並び替え結果に基づいて、名前を表示する
63 for (i = 0; i < 10; i++) {
64   // 例) i = 1 のとき s には 91 点が入っている
65   // 名前も運動して並び替えられると良いが、
66   // そういう作りにはしていないので、
67   // backup配列を参照して、
68   // 91 点の人は何番目であったか、
69   // 検索する
70   s = score[i];
71   for (j = 0; j < 10; j++) {
72     if (s == backup[j]) {
73       // 同じ点数（奈美子と太郎）の場合、太郎が表示されなくなることを防ぐため、
74       // あり得ない点数の -1 を設定する。
75       // （「番兵」と呼ばれる）
76       backup[j] = -1;
77       break; // 元の順番では、j 番目であったことが分かった
78     }
79   }
80   printf("%d %s\n", score[i], name[j]);
81 }
82
83 return 0;
84 }
```

♣ 素数

```

1 ****
2 * 1と自分自身以外で割り切れない数のことを「素数」と言います。
3 * 1～100までの間の素数を表示してみましょう。
4 * また1000個目の素数は何でしょうか？
5 ****
6 #include <stdio.h>
7
8 #define TRUE 1
9 #define FALSE 0
10
11 // 素数判定関数
12 int is_prime(int candidate, int prime_array[], int prime_index);
13
```

```

14 int main(int argc, char const *argv[]) {
15     // 素数の数 pi(x) ~= x / log(x) が知られている。
16     // なので10000までの素数の数は、たかだか2000 である。
17     // よって、要素数2000とする
18     int prime_array[2000] = {};
19
20     // = {} と書くことで、以下のように各要素を初期化したのと同等となる
21     // for (int i = 0; i < 2000; i++){
22     //     prime_array[i] = 0;
23     // }
24
25     int prime_index = 0;
26     int i;
27
28     // 2 は 偶数唯一の素数である。
29     prime_array[0] = 2;
30     prime_index = 1;
31
32     int candidate = 3; // 素数候補 3から始める
33
34     while (candidate < 10000) {
35         // 素数かどうかの判定は、is_prime関数に任せ、
36         // 素数であったら、配列に追加する
37         if (is_prime(candidate, prime_array, prime_index)) {
38             prime_array[prime_index] = candidate;
39             prime_index++;
40             candidate += 2; // 奇数の候補のみ調べるので、+2 している
41         } else {
42             candidate += 2;
43         }
44     }
45
46     // 結果表示
47     // (せっかくなので、10000までの素数表示)
48     for (int i = 0; i < prime_index; i++) {
49         if (i % 10 == 0) {
50             printf("%5d: ", i + 1);
51         }
52         if (prime_array[i] != 0) {
53             printf("%5d ", prime_array[i]);
54         }
55         // 10 個 ずつ 改行
56         if ((i + 1) % 10 == 0) {
57             printf("\n");
58         }
59     }
60     printf("\n");
61
62     printf("1000番目の素数は、%d です。\\n", prime_array[999]);
63 }
64
65 // 素数判定関数
66 int is_prime(int candidate, int prime_array[], int prime_index) {

```

```

67 int i;
68 int precious_metal; // 素数でないにしても、貴金属ではある。
69
70 // 10000(=100*100) が 素数かどうかを調べるには、
71 // 100までの素数で割りきれるかどうか、確認すれば良い。
72 // ここでは簡単化のために、今までに判明している全ての素数で割り切れるか確認する。
73 for (i = 0; i < prime_index; i++) {
74     precious_metal = prime_array[i]; // 割り切れるかどうか
75     if (candidate % precious_metal == 0) {
76         // 割り切れたなら、素数ではない。
77         return FALSE;
78     }
79 }
80 return TRUE;
81 }
```

♣ 完全数

```

1 ****
2 * 完全数とは次のような数のことです。
3 * 6は、1でも2でも3でも割り切れます。そして  $6 = 1 + 2 + 3$  です。
4 * 28は、1と2と4と7と14で割り切れます。
5 * 1と2と4と7と14を足すと28になります。
6 * 28の次の完全数は、いくつでしょうか？
7 ****
8 #include <stdio.h>
9
10#define TRUE 1
11#define FALSE 0
12
13// 完全数判定関数
14int is_perfect(int candidate);
15
16int main(int argc, char const *argv[]) {
17    // 6 = 2 * 3
18    // 28 =  $2^2 * 7$  である
19    // 割り切れるかどうかを調べ、
20    // 割り切った合計が、もとの数と一致するか調べればよい。
21
22    int perfect_array[4] = {6, 28};
23    int perfect_index = 2;
24
25    int candidate = 28 + 2; // 奇数の完全数は知られていないため、
26                           // 28の次の偶数を候補とする
27
28    while (1) {
29        // 完全数かどうかの判定は、is_perfect関数に任せ、
30        // 完全数が発見されるまで、繰り返す
31        if (is_perfect(candidate)) {
32            // 完全数が発見されたら追加
33            perfect_array[perfect_index] = candidate;
34            perfect_index++;
```

```

35     if (perfect_index == 4) {
36         break;
37     } else {
38         candidate += 2;
39     }
40     } else {
41         candidate += 2;
42     }
43 }
44
45 // 結果表示
46 for (int i = 0; i < perfect_index; i++) {
47     printf(" %d 番目の完全数は、%5d です。\\n", i + 1, perfect_array[i]);
48 }
49 }
50
51 // 完全数判定関数
52 int is_perfect(int candidate) {
53     int sum = 0; // 総和
54     int i;
55
56     for (i = 1; i < candidate; i++) {
57         if (candidate % i == 0) {
58             sum += i;
59         }
60     }
61     if (sum == candidate) {
62         return TRUE;
63     } else {
64         return FALSE;
65     }
66 }
```

♣ 英単語帳アプリ

```

1 ****
2 * 英単語帳アプリを創ってみましょう。
3 * I -> わたし
4 * love -> 愛する
5 * you -> あなた
6 * と答えられたら、満点です。
7 * 10問出題して、英語力向上を目指しましょう。
8 ****
9
10 #include "mt.h"
11 #include <stdio.h>
12
13 #define CHOICES_SIZE 3 // 三択で出題する
14 #define TRUE 1
15 #define FALSE 0
16
17 int main(int argc, char const *argv[]) {
```

```

18 // 出題用配列
19 char *english_words[] = {"", "I", "love", "you",
20                 "C", "language", "lesson", "happy",
21                 "hacker", "programming", "computer"};
22 char *japanese_words[] = {"", "わたし", "愛する", "あなた",
23                 "C", "言語", "学習", "幸せ",
24                 "技術者", "プログラミング", "コンピュータ"};
25
26 // 三択で出題する
27 int choices[CHOICES_SIZE + 1]; // 0 は未使用とするため、+1
28 int candidate; // 選択肢の候補
29 int registered_word = 10; // 登録単語数
30 int question_number; // 第何問目か？
31 int correct_answer; // 正答
32 int your_answer; // 入力された回答
33 int score; // 得点
34 int flag;
35 int i;
36
37 // オープニング
38 printf("*****\n");
39 printf("*\n");
40 printf("*      英単語ゲームへようこそ\n");
41 printf("*\n");
42 printf("*****\n");
43 printf("\n");
44
45 // 出題処理
46
47 // 10題出題する
48 score = 0;
49 for (question_number = 1; question_number <= 10; question_number++) {
50     printf("【第 %d 問】\n", question_number);
51
52     // 選択肢の初期化
53     for (i = 0; i <= CHOICES_SIZE; i++) {
54         choices[i] = 0;
55     }
56
57     // choices[1], [2], [3] に出題番号をセット
58     do {
59         // 1から3の乱数を取得
60         candidate = genrand_int32() % registered_word + 1;
61         // 選択肢に登録済みか調べる。
62         int flag = FALSE;
63         for (i = 1; i <= CHOICES_SIZE; i++) {
64             if (candidate == choices[i]) {
65                 flag = TRUE; // すでに選ばれていた
66                 break;
67             }
68         }
69         // 選択肢には未登録だったので、登録する
70         if (flag == FALSE) {

```

```

71     for (i = 1; i <= CHOICES_SIZE; i++) {
72         if (choices[i] == 0) {
73             // i番目の選択肢として登録する
74             choices[i] = candidate;
75             break;
76         }
77     }
78 }
79 // choice[3] ( 最後の選択肢 ) に0以外の値がセットされるまで繰り返す
80 } while (choices[CHOICES_SIZE] == 0);
81
82 // choice[1], [2], [3] の中から、いずれかを正解として設定する
83 correct_answer = genrand_int32() % 3 + 1;
84
85 // 出題する
86 printf("%s\n", english_words[choices[correct_answer]]);
87
88 // 選択肢を提示する
89 for (i = 1; i <= CHOICES_SIZE; i++) {
90     printf("%d: %s ", i, japanese_words[choices[i]]);
91 }
92 printf("\n");
93
94 // 1-3までの入力を求める
95 do {
96     scanf("%d", &your_answer);
97     while (getchar() != '\n')
98         ; // キーバッファ読み飛ばす
99 } while (your_answer <= 0 || your_answer > CHOICES_SIZE);
100
101 // 正解判定
102 if (your_answer == correct_answer) {
103     printf("正解です！\n\n");
104     score++; // 得点加算
105 } else {
106     printf("残念。正解は、%d: %s です。\n\n", correct_answer,
107           japanese_words[choices[correct_answer]]);
108 }
109 }
110
111 // 総合結果発表
112 printf("*****\n");
113 printf("*          結 果 発 表          *\n");
114 printf("*          10 問 中 %d 問 正解\n", score);
115 printf("*          おめでとうございます！          *\n");
116 printf("*****\n");
117 printf("\n");
118
119 return 0;

```

124 }

3.5 作成例 ★★★★★

♣ 生きてきた日数

```

1  ****
2  * 1980年1月1日生まれの人は、今日までに何日生きたことになるでしょうか？
3  * (生年月日は 19800101 と入力されます。)
4  * 昭和20年8月10日生まれの方は、どうでしょうか？
5  * (生年月日は 3200810 と入力されます。)
6  * 生後30000日目を迎えるのは、何年何月何日でしょうか？
7  ****
8
9 #include <math.h>
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13 #include <time.h> // 日付時刻の操作用関数
14
15 // 年月日からユリウス通日を求める関数
16 int julian_day_number(int year, int month, int day);
17 // ユリウス通日から年月日を返す関数
18 void inverse_julian_day_number(int jdn, int *year, int *month, int *day);
19
20 int main(int argc, char const *argv[]) {
21     // コマンドラインからの引数がなければ、使い方表示
22     if (argc == 1) {
23         printf("【使い方】\n");
24         printf("今日まで何日来たかを表示します。\n");
25         printf("1980年1月1日生まれであれば、\n");
26         printf("%s 19800101\n", argv[0]);
27         printf("昭和20年8月10日生まれであれば、\n");
28         printf("%s 3200810\n", argv[0]);
29         printf("の様に入力して下さい。\n");
30         exit(1);
31     }
32
33     int birthday; // 生年月日
34     char *endptr; // 数値に変換出来なかった文字
35     int b_year; // 誕生日の西暦年
36     int b_month; // 誕生日の月
37     int b_day; // 誕生日の日
38
39     // 簡易エラーチェック
40     // 数値変換不可の文字列の長さを調べる。
41     birthday = strtol(argv[1], &endptr, 10); // 10進数として変換する
42     if (strlen(endptr) != 0) {
43         // エラーメッセージ出力
44         printf("%s は、生年月日として認識出来ませんでした。\n", endptr);
45         exit(1);
46     }
47
48     // 生年月日に分離する

```

```
49 // 年
50 // 元号入力なら、西暦へ変換
51 if (birthday < 1e7) {
52     int ge = birthday / 10000; // 上3桁 元号符号+元号年
53     switch (ge / 100) {
54         case 1:
55             b_year = ge % 100 + 1867;
56             break;
57         case 2:
58             b_year = ge % 100 + 1911;
59             break;
60         case 3:
61             b_year = ge % 100 + 1925;
62             break;
63         case 4:
64             b_year = ge % 100 + 1988;
65             break;
66     }
67 } else {
68     b_year = birthday / 10000; // 上4桁は西暦年
69 }
70 // 月
71 b_month = birthday / 100; // 下二桁を切り捨てて
72 b_month = b_month % 100; // 残った数の下2桁
73
74 // 日
75 b_day = birthday % 100; // 下2桁は日
76
77 // http://simd.jugem.jp/?eid=149 より引用
78 // 今日の日付を取得する
79 time_t now;
80 struct tm *ltm;
81 time(&now);
82 ltm = localtime(&now);
83
84 // 確認用
85 // printf( "%5d : [年]\n", ltm->tm_year + 1900 );
86 // printf( "%5d : [月]\n", ltm->tm_mon + 1 );
87 // printf( "%5d : [日]\n", ltm->tm_mday );
88 // printf( "%5d : [時]\n", ltm->tm_hour );
89 // printf( "%5d : [分]\n", ltm->tm_min );
90 // printf( "%5d : [秒]\n", ltm->tm_sec );
91 // printf( "%5d : [曜日]\n", ltm->tm_wday );
92 // printf( "%5d : [経過日数]\n", ltm->tm_yday );
93 // printf( "%5d : [夏時間の有無]\n", ltm->tm_isdst );
94
95 int t_year = ltm->tm_year + 1900; // 今日の西暦年
96 int t_month = ltm->tm_mon + 1; // 今日の月
97 int t_day = ltm->tm_mday; // 今日の日
98
99 // strftime関数を用いて文字列にすることも可能
100 // char str_time[100];
101 // int maxsize = 100;
```

```

102 // char *format = "%Y年%m月%d日 %H:%M";
103 // strftime(str_time, maxsize, format, ltm);
104 // printf("%s\n", str_time);
105
106 // まともに暦の日付計算をしようとすると大変なので、
107 // ある基準日からの経過日数を求めることにします。
108 // 今日は、 基準日から何日目
109 // 誕生日は、基準日から何日目と分かれば、
110 // 引き算することで、日数を計算出来ます。
111 // ユリウス通日(つうじつ)として知られており、
112 // 紀元前4713年1月1日を第一日として、
113 // 天文学などの分野で用いられています。
114 // https://ja.wikipedia.org/wiki/ユリウス通日
115 // に公式がありますので、プログラミングしましたが、
116 // http://ufcpp.net/study/algorithm/o\_days.html
117 // にも説明があります。
118
119 printf("今日は      %4d 年 %2d 月 %2d 日です。\\n", t_year, t_month, t_day);
120 printf("誕生日は      %4d 年 %2d 月 %2d 日です。\\n", b_year, b_month, b_day);
121
122 // 今日のユリウス通日
123 int today_jdn = julian_day_number(t_year, t_month, t_day);
124 // printf("今日の ユリウス通日 %d\\n", today_jdn);
125 int birthday_jdn = julian_day_number(b_year, b_month, b_day);
126 // printf("誕生日の ユリウス通日 %d\\n", birthday_jdn);
127
128 // 結果表示
129 printf("今日は生後 %5d 日 です。\\n", today_jdn - birthday_jdn);
130
131 // 生まれてから30000日後がいつかを求めます。
132 int life_30000_jdn = birthday_jdn + 30000;
133 int l_year, l_month, l_day;
134 // l_year, l_month, l_day に値がセットされるよう、アドレスを渡します。
135 inverse_julian_day_number(life_30000_jdn, &l_year, &l_month, &l_day);
136 printf("生後三万日は %4d 年 %2d 月 %2d 日 です。\\n", l_year, l_month, l_day);
137
138 return 0;
139 }
140
141 // 年月日からユリウス通日を求める関数
142 int julian_day_number(int year, int month, int day) {
143 /*
144     https://ja.wikipedia.org/wiki/ユリウス通日
145     // 2月のユリウス通日が不正
146
147     int y, m, d;
148     int n;
149     int mjd;
150     int jdn;
151
152     y = year + (month - 3) / 12;
153     m = (month - 3) % 12;
154     d = day - 1;

```

```

155     n = d + (153*m+2)/5 + 365*y + y/4 - y/100 + y/400;
156     mjd = n - 678881;
157     jdn = mjd + 2400001;
158 */
159
160     int a = (14 - month) / 12;
161     int y = year + 4800 - a;
162     int m = month + 12 * a - 3;
163
164     int jdn =
165         day + (153 * m + 2) / 5 + 365 * y + y / 4 - y / 100 + y / 400 - 32045;
166
167     return jdn;
168 }
169
170 // 年月日からユリウス通日を返す関数
171 void inverse_julian_day_number(int jdn, int *year, int *month, int *day) {
172     // 複数の値を返すときは、ポインタで返します
173
174 /*
175     int n = jdn - 2400001 + 678881;
176     printf("inv_jdn: n: %d\n", n);
177     int a = 4*n + 3 + 4*floor(3.0/4.0 * (floor(4*(n+1)/146097) + 1));
178     int b = 5 * floor((a%1461)/4) + 2;
179
180     int y = a / 1461;
181     int m = b / 153;
182     int d = (b % 153) / 5;
183
184     // y = year + (month - 3) / 12;
185     // m = (month - 3) % 12;
186     // d = day - 1;
187
188     // であるから、
189     day = d + 1;
190     month = (m + 3);
191     if (month > 12) {
192         month = month - 1;
193     }
194 */
195
196 // https://en.wikipedia.org/wiki/Julian_day
197 int f = jdn + 1401 + (((4 * jdn + 274277) / 146097) * 3) / 4 - 38;
198
199 int e = 4 * f + 3;
200 int g = (e % 1461) / 4;
201 int h = 5 * g + 2;
202 int D = (h % 153) / 5 + 1;
203 int M = (h / 153 + 2) % 12 + 1;
204 int Y = (e / 1461) - 4716 + (12 + 2 - M) / 12;
205
206 *year = Y;
207 *month = M;

```

```
208     *day = D;
209 }
```

♣ カレンダー

```

1 ****
2 * 年と月を入力すると、
3 * その月のカレンダーを表示するプログラムを作ってみましょう。
4 * 「ツェラーの公式」を用いると曜日を求めることが出来ます。
5 * (西暦1年1月1日は月曜日となりますので、
6 * 7で割った余りを求めてことで、曜日が計算出来ます)
7 ****
8
9 #include <stdio.h>
10 #include <stdlib.h>
11 #include <string.h>
12
13 // ツェラーの公式(Zeller's congruence)により 曜日を返す関数
14 // 戻り値が 0, 1, 2, 3, 4, 5, 6 の場合、
15 // それぞれ 日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日
16 int zeller(int year, int month, int day);
17
18 // 今月が何日まであるかを返す
19 int last_day_of_month(int year, int month);
20
21 int main(int argc, char const *argv[]) {
22     //
23     // コマンドラインからの引数がなければ、使い方表示
24     //
25     if (argc == 1) {
26         printf("【使い方】\n");
27         printf("カレンダーを表示します。\n");
28         printf("2016年 6月のカレンダーであれば\n");
29         printf("%s 2016 6\n", argv[0]);
30         printf("の様に入力して下さい。\n");
31         exit(1); // プログラム終了
32     }
33
34     //
35     // 変数宣言
36     //
37     int year; // 西暦年
38     int month; // 月
39     int day; // 日(カレンダー表示用)
40
41     char *err_str; // 数値に変換出来なかった文字
42
43     int day_of_week; // 曜日
44     int week_number; // 月の第何週か？
45     char *days_name[] = {"日", "月", "火", "水", // 曜日の名前
46                           "木", "金", "土"};
```

```
48 int last_day; // 今月は何日が最後の日か ?
49
50 int h; // 今月1日が何曜日であるか ? (ツェラーの公式)
51
52 //
53 // 簡易エラーチェック
54 //
55
56 // 数値変換不可の文字列の長さを調べる。
57 year = strtol(argv[1], &err_str, 10); // 10進数として変換する
58 if (strlen(err_str) != 0) {
59     // エラーメッセージ出力
60     printf("%s は、西暦年として認識出来ませんでした。\\n", err_str);
61     exit(1); // プログラム終了
62 }
63 // 数値変換不可の文字列の長さを調べる。
64 month = strtol(argv[2], &err_str, 10); // 10進数として変換する
65 if (strlen(err_str) != 0) {
66     // エラーメッセージ出力
67     printf("%s は、月として認識出来ませんでした。\\n", err_str);
68     exit(1); // プログラム終了
69 }
70
71 //
72 // カレンダー表示処理
73 //
74
75 // カレンダーの年月と曜日名を表示
76 printf("\\n %4d 年 %2d 月\\n", year, month);
77 for (day_of_week = 0; day_of_week <= 6; day_of_week++) {
78     printf(" %s", days_name[day_of_week]);
79 }
80 printf("\\n");
81
82 // ツェラーの公式で、その月の1日が何曜日始まりであるか、取得する。
83 // 戻り値が 0, 1, 2, 3, 4, 5, 6 の場合、
84 // それぞれ 日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日
85 h = zeller(year, month, 1);
86
87 // 今月の最終日を求める
88 last_day = last_day_of_month(year, month);
89
90 // 水曜日始まりの月の場合、
91 // 日、月、火 の欄に日付を表示させたくないの、
92 // day = 1 - h からスタートする
93 day = 1 - h;
94 day_of_week = 0;
95 do {
96     if (day < 1) {
97         printf("   "); // 1日以前なら空欄を出力
98         day++;
99     } else {
100        printf(" %2d", day);
```

```

101     day++;
102 }
103 day_of_week++;
104 if (day_of_week % 7 == 0) {
105     printf("\n"); // 週送り
106 }
107 } while (day <= last_day);
108 printf("\n");
109
110 return 0;
111 }

112 // ツェラーの公式(Zeller's congruence)により 曜日を返す関数
113 // 戻り値が 0, 1, 2, 3, 4, 5, 6 の場合、
114 // それぞれ 日曜日、月曜日、火曜日、水曜日、木曜日、金曜日、土曜日
115 int zeller(int year, int month, int day) {
116     int y, m, d;
117     int h;
118
119     // 年月日の設定
120     // 但し1月、2月は前年の13月、14月として計算
121     if (month == 1 || month == 2) {
122         y = year - 1;
123         m = month + 12;
124         d = day;
125     } else {
126         y = year;
127         m = month;
128         d = day;
129     }
130 }
131
132 // 曜日の算出
133 h = (y + y / 4 - y / 100 + y / 400 + (13 * m + 8) / 5 + d) % 7;
134
135 return h;
136 }

137 // 今月が何日まであるかを返す
138 // int ldom[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
139 // のように配列に値を持たせても良い。
140 int last_day_of_month(int year, int month) {
141     switch (month) {
142     case 2:
143         if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
144             return 29;
145         } else {
146             return 28;
147         }
148     case 4:
149     case 6:
150     case 9:
151     case 11:
152         return 30;
153     }

```

```

154
155     case 1:
156     case 3:
157     case 5:
158     case 7:
159     case 8:
160     case 10:
161     case 12:
162         return 31;
163     default:
164         return 0;
165     }
166 }
```

♣ 双六ゲーム

```

1 ****
2 * 双六ゲームを創ってみましょう。
3 * 止まった升目に「3つ進む」や、「振り出しに戻る」も創ってみましょう。
4 * どこまで進んだか分かる表示機能や、
5 * オープニング・エンディングもあると楽しいですね。*
6 ****
7
8 #include "mt.h"
9 #include <stdio.h>
10 #include <string.h>
11
12 #define MAP_SIZE 22      // 0-21までの升目がある
13 #define GOAL_POSITION 21 // 21の升目が、上がり
14 #define TRUE 1
15 #define FALSE 0
16
17 // 双六のマップ配置
18 // 0: スタート
19 // 1:
20 // 2:
21 // 3:
22 // 4: 2マス進む
23 // 5: 3マス戻る
24 // 6:
25 // 7: スタートに戻る
26 // 8: 2マス進む
27 // 9: 1回休み
28 // 10: 3マス戻る
29 // 11:
30 // 12: 2マス進む
31 // 13:
32 // 14: スタートに戻る
33 // 15: 3マス戻る
34 // 16: 2マス進む
35 // 17:
36 // 18: 2回休み
```

```

37 // 19 :
38 // 20 : 3マス戻る
39 // 21 : ゴール
40
41 // main の外側に配置(グローバル変数)して、各関数から共通して見られるようにする。
42 // (一般的には好ましくないが、双六作成が容易となるため、許容する)
43 char const *map_event[] = {"", "", "", "", "2A", "3B", "", "SB",
44                         "", "1R", "3B", "", "2A", "", "SB", "3B",
45                         "2A", "", "2R", "", "3B", ""};
46
47 // 列挙型 cmp(競技者)型の宣言
48 typedef enum { PLAYER, COMPUTER } cmp;
49 // cmp型変数 competitorの宣言
50 cmp competitor;
51 // #define PLAYER 0
52 // #define COMPUTER 1 と同じ
53 // enum のご紹介のみ
54
55 // PLAYER, COMPUTER それぞれが、双六上のどの升目にいるか？
56 int position[2] = {};
57 // PLAYER, COMPUTER 何回休みか？
58 char rest[2];
59
60 // 関数のプロトタイプ宣言 省略するため
61 // main 関数を末尾に記載
62
63 // 双六表示機能
64 void draw_map() {
65     int i;
66     printf(
67         ""
68     );
69     printf(
70         " ST 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 GL \n");
71     printf(
72         "+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+\n");
73     // player
74     for (i = 0; i < MAP_SIZE; i++) {
75         printf("| ");
76         if (position[PLAYER] == i) {
77             printf("P");
78         } else {
79             printf(" ");
80         }
81     }
82     printf("\n");
83     // computer
84     for (i = 0; i < MAP_SIZE; i++) {
85         printf("| ");
86         if (position[COMPUTER] == i) {
87             printf("C");
88         } else {
89             printf(" ");
90         }

```

```

90    }
91    printf("|\n");
92    // event
93    printf(
94        "+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+\n");
95    printf(
96        " | | | | 2A|3B| |SB| |1R|3B| |2A| |SB|3B|2A| |2R| |3B| |\n");
97    printf(
98        "+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+\n");
99    printf(
100        "
101        \n");
102    printf(
103        " 【凡例】A: 進む B: 戻る R: 休み S:振り出し
104        "
105        \n");
106
107 void draw_opening() {
108     int i;
109     for (i = 0; i < 24; i++) {
110         puts("");
111     }
112     puts("#####");
113     puts("#");
114     puts("# The Japanese Traditional Game");
115     puts("#");
116     puts("# ##### # # ##### ##### ##### # # # # #");
117     puts("# # # # # # # # # # # # # # # # # # #");
118     puts("# # # # # # # # # # # # # # # # # # #");
119     puts("# ##### # # # # ##### # # ##### # # #");
120     puts("# # # # # # # # # # # # # # # # # # #");
121     puts("# # # # # # # # # # # # # # # # # # #");
122     puts("# ##### ##### # # ##### # # ##### # #");
123     puts("#");
124     puts("#####");
125     puts("");
126 }
127
128 void draw_endng() {
129     puts("");
130     puts("#####");
131     puts("#");
132     puts("# The Japanese Traditional Game");
133     puts("#");
134     puts("# S U G O R O K U");
135     puts("#");
136     puts("# Fin.");
137     puts("#");
138     puts("#####");
139     puts("");
140 }
141
142 // マップ上のイベントを受け取り、競技者の内部状態を適宜変更する

```

```

143 // " | | | | |2A|3B| |SB| |1R|3B| |2A| |SB|3B|2A| |2R| |3B| |\n";
144 // legend: nV. n: times V: verb. go Ahead / go Backward / Rest \n");
145 void event(cmp competitor) {
146     if (position[competitor] > GOAL_POSITION) {
147         position[competitor] = GOAL_POSITION;
148     }
149
150     // イベントを取得
151     char event[3];
152     strcpy(event, map_event[position[competitor]]);
153     if (strlen(event) == 0) {
154         // 何もイベントはなかったとして終了
155         return;
156     }
157
158     // 該当イベントの処理
159     char times = event[0]; // 何升進む、何回休みなどの数
160     char kind = event[1]; // 進む、戻る、休むの種類
161     switch (kind) {
162         case 'A': // 進む
163             if (times == 'S') {
164                 // 振り出しに戻る
165                 position[competitor] = 0;
166             } else {
167                 // 進める
168                 position[competitor] += (times - '0'); // 文字を数値に変換
169                 // ゴールを通り過ぎていたら、ゴール地点に調整
170                 if (position[competitor] > GOAL_POSITION) {
171                     position[competitor] = GOAL_POSITION;
172                 }
173             }
174             // メッセージ表示
175             if (competitor == PLAYER) {
176                 printf("P> やった～ %d マス 進んだ！\n", times - '0');
177             } else {
178                 printf("C> やった～ %d マス 進んだ！\n", times - '0');
179             }
180             break;
181         case 'B': // 戻る
182             if (times == 'S') {
183                 // 振り出しに戻る
184                 position[competitor] = 0;
185             } else {
186                 // 戻る
187                 position[competitor] -= (times - '0'); // 文字を数値に変換
188                 // スタートを通り過ぎていたら、スタート地点に調整
189                 if (position[competitor] < 0) {
190                     position[competitor] = 0;
191                 }
192             }
193             // メッセージ表示
194             if (times == 'S') {
195                 if (competitor == PLAYER) {

```

```

196     printf("P> わ～ん スタートに 戻ったよ\n");
197 } else {
198     printf("C> わ～ん スタートに 戻ったよ\n");
199 }
200 } else {
201     if (competitor == PLAYER) {
202         printf("P> わ～ん %d マス 戻ったよ\n", times - '0');
203     } else {
204         printf("C> わ～ん %d マス 戻ったよ\n", times - '0');
205     }
206 }
207 break;
208 case 'R':           // お休み
209     rest[competitor] = (times - '0'); // 文字を数値に変換
210 // メッセージ表示
211 if (competitor == PLAYER) {
212     printf("P> わ～ん %d 回 休みだよ\n", times - '0');
213 } else {
214     printf("C> わ～ん %d 回 休みだよ\n", times - '0');
215 }
216 break;
217 }
218 }
219 // さいころを振って進む
220 void dice_and_walk(cmp competitor) {
221     int dice = genrand_int32() % 6 + 1;
222     position[competitor] += dice;
223 // ゴールを通り過ぎていたら、ゴール地点に調整
224     if (position[competitor] > GOAL_POSITION) {
225         position[competitor] = GOAL_POSITION;
226     }
227 // メッセージ表示
228     if (competitor == PLAYER) {
229         printf("P> やった～ %d マス 進んだ！\n", dice);
230     } else {
231         printf("C> やった～ %d マス 進んだ！\n", dice);
232     }
233 }
234 }
235 // 双六を上がったか、判定関数
236 int is_goal(cmp competitor) {
237     if (position[competitor] == GOAL_POSITION) {
238         return TRUE;
239     } else {
240         return FALSE;
241     }
242 }
243 }
244
245 int main(int argc, char const *argv[]) {
246 // タイトル表示
247 draw_opening();

```

```

249
250 //双六表示
251 draw_map();
252
253 printf("\nPress Enter key\n");
254 while (getchar() != '\n')
255 ;
256
257 // 競技開始
258 // どちらかが上がるまで、続行
259 do {
260
261     // PLAYERの番
262     printf("\nPLAYERの番 Press Enter key\n");
263     while (getchar() != '\n')
264         ;
265
266     if (rest[PLAYER] == 0) {
267         // ○回休みでなければ、さいころを振って進む
268         dice_and_walk(PLAYER);
269         // 止まった升目にないかイベントが設定されているか
270         event(PLAYER);
271     } else {
272         // お休み回数を減らす
273         printf("P> %d 回休みなので進めない・・・\n", rest[PLAYER]);
274         rest[PLAYER]--;
275     }
276
277     //双六表示
278     draw_map();
279
280     // COMPUTERの番
281     printf("\nCOMPUTERの番 Press Enter key\n");
282     while (getchar() != '\n')
283         ;
284
285     if (rest[COMPUTER] == 0) {
286         // ○回休みでなければ、さいころを振って進む
287         dice_and_walk(COMPUTER);
288         // 止まった升目にないかイベントが設定されているか
289         event(COMPUTER);
290     } else {
291         // お休み回数を減らす
292         printf("C> %d 回休みなので進めない・・・\n", rest[COMPUTER]);
293         rest[COMPUTER]--;
294     }
295
296     //双六表示
297     draw_map();
298
299 } while (!is_goal(PLAYER) && !is_goal(COMPUTER));
300
301 //ゴール到着時のメッセージ

```

```

302     if (is_goal(PLAYER)) {
303         printf("P> 双六を上がったよ(*^_^*)\n");
304     } else {
305         printf("C> 双六を上がったよ(*^_^*)\n");
306     }
307
308     // エンディング表示
309     draw_ending();
310
311     return 0;
312 }
```

♣ 数当てゲーム (Match Number)

```

1  ****
2  * 数当てゲーム Match Number
3  * 事前に用意された3桁の数字を、ヒントをもとに当てていくゲームです。
4  * 用意された数字が 9 2 5 だとします。
5  * 9 9 9と入れると、9は正解ですので、1つ正解と表示します。
6  * 5 2 0と入れると、5と2は正解ですので、2つ正解と表示します。
7  * 5 9 2と入れると、(順番は違いますが) 3つとも合っていますので、
8  * 3つ正解と表示します。
9  * これを繰り返すことで、事前に用意された数字、9 2 5を当てるゲームです。
10 ****
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <string.h>
15
16 int main(int argc, char const *argv[]) {
17     int right_number[] = {9, 2, 5}; // 事前に用意された数
18                                         // 955など重複した数は対象外
19     char your_number[8];           // 入力された数字
20     int number;
21     int unique_number[3]; // 999の入力なら 9
22                           // 990の入力なら 9, 0
23                           // 987の入力なら 9, 8, 7
24
25     int score; // いくつ合っていたか
26     int i, j; // loop counter
27
28     // 一回目の入力処理
29     // while文内の処理と重複するが、許容する。
30     printf("三桁の数字を入力して下さい\n");
31     printf("終了するには、\"quit\"と入力して下さい。.\n");
32     printf("\n");
33     scanf("%s", your_number);
34     while (getchar() != '\n')
35     ;
36
37     // "quit"が入力されるまで繰り返す
38     while (strcmp(your_number, "quit")) {
```

```

39
40 // 入力文字'n'を数nに変換
41 for (i = 0; i < 3; i++) {
42     unique_number[i] = your_number[i] - '0'; //文字'n'を数nへ変換
43 }
44
45 for (i = 0; i < 3; i++) {
46     printf("0' unique_number[%d]: %d ", i, unique_number[i]);
47 }
48 printf("\n");
49
50 // 仮に 990 と入力されているなら、90 をMatch Number の対象にするので、
51 // 重複を排除
52 for (i = 2; i >= 1; i--) { // un[2]は、un[1], un[0]と等しいか？
53 // un[1]は、un[0]と等しいか？
54     for (j = i - 1; j >= 0; j--) {
55         printf("unique[%d]:%d unique[%d]:%d \n", i, unique_number[i], j,
56             unique_number[j]);
57         if (unique_number[i] == unique_number[j]) {
58             unique_number[i] = -1; // 自分より小さい添字の要素が、
59 // 自分自身と重複していることが判明したため、
60 // 未使用の意味で、-1をセットする
61         }
62     }
63 }
64
65 // いくつ合っているか、出力
66 score = 0;
67 for (i = 0; i < 3; i++) {
68     if (unique_number[i] != -1) {
69         number = unique_number[i];
70     } else {
71         break;
72     }
73     for (j = 0; j < 3; j++) {
74         if (number == right_number[j]) {
75             score++;
76             break;
77         }
78     }
79 }
80 printf("正解数: %d \n\n", score);
81
82 for (i = 0; i < 3; i++) {
83     printf("unique_number[%d]: %d ", i, unique_number[i]);
84 }
85 printf("\n");
86 for (i = 0; i < 3; i++) {
87     printf("right_number[%d]: %d ", i, right_number[i]);
88 }
89 printf("\n");
90
91 // 入力された数値の配列と、正解の配列が等しいことを確認する

```

```

92     if (unique_number[0] == right_number[0] &&
93         unique_number[1] == right_number[1] &&
94         unique_number[2] == right_number[2]) {
95             // if(memcmp(unique_number, right_number, sizeof(unique_number)) == 0){ //
96             // こう書くことも出来ます。
97             printf("おめでとうございます。正解です！！\n");
98             break;
99         } else {
100             printf("三桁の数字を入力して下さい\n");
101             printf("終了するには、\"quit\"と入力して下さい。\n");
102             printf("\n");
103             scanf("%s", your_number);
104             while (getchar() != '\n')
105                 ;
106         }
107     } // while end
108 }
```

♠ ポーカー

```

1 /*=====
2 /*
3 /* ポーカー
4 /*
5 /* https://ja.wikipedia.org/wiki/ポーカー・ハンドの一覧 */
6 /* https://simple.wikipedia.org/wiki/Poker#Hands */
7 =====*/
8
9 #include <stdio.h>
10 #include <stdlib.h>
11 #include <string.h>
12 #include <time.h>
13
14 #define BUFFER_SIZE 255 // 文字列用のバッファサイズ
15
16 // printfを簡単にするためのマクロ定義
17 #define prd(dt) printf(#dt ":%d\n", dt)
18 #define prs(dt) printf(#dt ":%s\n", dt)
19 #define prx(dt) printf(#dt ":%x\n", dt)
20
21 // カードの種類
22 typedef enum { CLUBS = 4, DIAMONDS, HEARTS, SPADES } suits;
23 typedef enum {
24     NO_PAIR,           // 役無し
25     ONE_PAIR,          // ワンペア
26     TWO_PAIR,          // ツーペア
27     THREE_OF_A_KIND,  // スリーカード
28     STRAIGHT,          // ストレート
29     FLUSH,             // フラッシュ
30     FULL_HOUSE,        // フルハウス
31     FOUR_OF_A_KIND,   // フォーカード
32     STRAIGHT_FLUSH,   // ストレートフラッシュ
33 }
```

```

33         // HANDS_COUNT,      //
34 } hands_type;
35
36 // プロトタイプ宣言
37 void init_card();
38 void init_player_hands(char player_hands[3][6]);
39 void init_score();
40
41 void disp_hands();
42 void disp_score();
43 void disp_score();
44
45 void disp_card(char mycard);
46
47 int set_card(char mycard, int player);
48 void calc_card();
49 char get_card();
50 void poker_hands(int player);
51 int my_random(int n);
52 void pause();
53 void usage(char const *argv[]);
54
55 int is_no_pair(int player);
56 int is_one_pair(int player, int seach);
57 int is_two_pair(int player);
58 int is_three_of_a_kind(int player);
59 int is_straight(int player);
60 int is_flush(int player);
61 int is_full_house(int player);
62 int is_four_of_a_kind(int player);
63 int is_suit(int player, int search);
64
65 // 文字コードとビット演算の学習を兼ねて、
66 // 以下のようにカードを表現する
67 ****
68 * A234567890JQK
69 * C ABCDEFGHIJKLMNOP
70 * D QRSTUVWXYZ[\]
71 * H abcdefghijklm
72 * Sqrstuvwxyz{|}
73 *
74 * 例 ) ABCTd と書くと
75 * C(クローバー)のA,2,3,
76 * D(ダイヤ)の4、
77 * H(ハート)の4でワンペア
78 ****
79
80 // player_hands[1]:EJKLM
81 // player_hands[2]:ejklm
82 // cards[6][17]
83 //          | A 2 3 4 5 6 7 8 9 0 J Q K A | 1 2
84 // -----+-----+-----+
85 // Clubs    | 0 0 0 0 1 0 0 0 0 1 1 1 1 0 | 13 0

```

```

86 // Diamonds | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0
87 // Hearts | 0 0 0 0 2 0 0 0 0 2 2 2 2 0 | 0 13
88 // Spades | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0
89 // -----+-----+
90 // Player 1 | 0 0 0 0 1 0 0 0 0 1 1 1 1 0 | 0 0
91 // Player 2 | 0 0 0 0 1 0 0 0 0 1 1 1 1 0 | 0 0
92 // プレーヤー1 の役は フラッシュです
93 // プレーヤー2 の役は フラッシュです
94
95 // グローバル変数
96 char cards[6][17]; // 1-K + A で14種類 player1/2の合計用に2ます追加
97 // 10,J,Q,K,A のストレート判定をしやすくするため、
98 // K の次に Aを設けている
99 int score[3][10][3]; // 役の強弱を判定するために用いる score[0] 未使用
100
101 int main(int argc, char const *argv[]) {
102     int i;
103     char ss[BUFFER_SIZE + 1]; // ファイルから一行読み込むための作業用
104     FILE *fp; // ファイルポインタ
105     int player; // 1ならプレーヤー1、2ならプレーヤー2
106     char mycard; // 一枚引いたカード
107     char player_hands[2 + 1][5 + 1]; // プレーヤーの手元にある5枚のカード
108 // 添字の0は未使用
109
110 // 初期化処理
111     init_card();
112     init_player_hands(player_hands);
113     init_score();
114
115 // コマンドライン引数なしなら、使い方表示して終了
116 if (argc < 2) {
117     usage(argv);
118     exit(1);
119 }
120
121 // -f ファイルからの読み取りオプションなら
122 if (strcmp(argv[1], "-f") == 0) {
123     if ((fp = fopen(argv[2], "r")) == NULL) {
124         // エラーメッセージ表示
125         printf("%s を開けませんでした。\\n", argv[2]);
126         exit(1);
127     }
128
129     player = 1;
130     while (1) {
131         // ファイルから手を読み込む
132         fgets(ss, BUFFER_SIZE, fp);
133         if (strchr(ss, '*') == NULL && strchr(ss, '/') == NULL) {
134             // コメント行でなければ、読み取った手をセットする
135             strcpy(player_hands[player], ss);
136             player_hands[player][5] = '\\0';
137             if (player == 2) {
138                 break;

```

```

139     } else {
140         player++;
141     }
142 }
143 }
144 fclose(fp);

145 // 各プレーヤーが手にしたカード情報を、カード管理配列へ書き出す
146 for (player = 1; player <= 2; player++) {
147     for (i = 0; i < 5; i++) {
148         mycard = player_hands[player][i];
149         // ファイル入力時は、同じカードは所有していないはずなので、
150         // エラーチェックはしていない
151         set_card(mycard, player);
152     }
153 }
154 }

155 // -m 交互入力オプションなら
156 } else {
157     for (player = 1; player <= 2; player++) {
158         for (i = 0; i < 5; i++) {
159             do {
160                 mycard = get_card(); // ランダムにカードを引く
161                 // prx(mycard); // 引いたカードを表示(debug)
162                 disp_card(mycard); // 引いたカードを表示
163                 pause();
164                 player_hands[player][i] = mycard;
165             } while (set_card(mycard, player) != 0); // 既に引いたカードなら、引き直す
166         }
167     }
168 }
169 }

170 }

171 // playerの手元にあるカードを表示
172 prs(player_hands[1]);
173 prs(player_hands[2]);

174 // 得点計算
175 calc_card();

176 // 役を計算
177 poker_hands(1); // player1 の役を計算
178 poker_hands(2); // player2 の役を計算
179

180 // カードの表示
181 disp_hands();

182 // 得点表示
183 disp_score();

184 // 役を表示
185 for (player = 1; player <= 2; player++) {
186     printf("プレーヤー%d の役は ", player);
187 }
```

```
192 switch (score[player][9][0] / 1000) {
193     case 8:
194         printf("ストレートフラッシュです\n");
195         break;
196     case 7:
197         printf("フォーカードです\n");
198         break;
199     case 6:
200         printf("フルハウスです\n");
201         break;
202     case 5:
203         printf("フラッシュです\n");
204         break;
205     case 4:
206         printf("ストレートです\n");
207         break;
208     case 3:
209         printf("スリーカードです\n");
210         break;
211     case 2:
212         printf("ツーペアです\n");
213         break;
214     case 1:
215         printf("ワンペアです\n");
216         break;
217     case 0:
218         printf("ふたです\n");
219         break;
220     }
221 }
222
223 // 勝敗表示
224 if (score[1][9][0] > score[2][9][0]) {
225     printf("プレーヤー1の勝ちです\n");
226 } else if (score[1][9][0] < score[2][9][0]) {
227     printf("プレーヤー2の勝ちです\n");
228 } else {
229     printf("引き分けです\n");
230 }
231 return 0;
232 }
233
234 // カード配列の初期化
235 void init_card() {
236     int i, j;
237     for (j = 0; j < 6; j++) {
238         for (i = 0; i < 17; i++) {
239             cards[j][i] = 0;
240         }
241     }
242 }
243
244 // 得点配列の初期化
```

```

245 void init_score() {
246     int i, j, player;
247     for (player = 0; player <= 2; player++) {
248         for (i = 0; i < 10; i++) {
249             for (j = 0; j < 3; j++) {
250                 score[player][i][j] = 0;
251             }
252         }
253     }
254 }
255
256 // プレーヤーの手にしているカードの初期化
257 void init_player_hands(char player_hands[3][6]) {
258     int i, j;
259     for (j = 0; j < 3; j++) {
260         for (i = 0; i < 6; i++) {
261             player_hands[j][i] = 0;
262         }
263     }
264 }
265
266 // カードの表示
267 void disp_hands() {
268     int i, j;
269     printf("\n");
270     printf("      | A 2 3 4 5 6 7 8 9 0 J Q K A |  1  2 \n");
271     printf("-----+-----+-----+-----\n");
272     for (j = 0; j < 6; j++) {
273         switch (j) {
274             case 0:
275                 printf("Clubs    | ");
276                 break;
277             case 1:
278                 printf("Diamonds | ");
279                 break;
280             case 2:
281                 printf("Hearts   | ");
282                 break;
283             case 3:
284                 printf("Spades   | ");
285                 break;
286             case 4:
287                 printf("Player 1 | ");
288                 break;
289             case 5:
290                 printf("Player 2 | ");
291                 break;
292         }
293         for (i = 1; i < 17; i++) {
294             if (i < 15) {
295                 printf("%1d ", cards[j][i]);
296             } else {
297                 printf("%2d ", cards[j][i]);

```

```
298     }
299     if (i == 14) {
300         printf(" | ");
301     }
302 }
303 printf("\n");
304 if (j == 3) {
305     printf("-----+-----\n");
306 }
307 }
308 printf("\n");
309 }
310
311 // 得点の表示
312 void disp_score() {
313     int i;
314     printf("      Player1      Player2\n");
315     printf("-----+-----\n");
316     for (i = 9; i >= 0; i--) {
317         switch (i) {
318             case 9:
319                 printf("<<MAX>> | ");
320                 break;
321             case 8:
322                 printf("S.Flush | ");
323                 break;
324             case 7:
325                 printf("4 cards | ");
326                 break;
327             case 6:
328                 printf("Full H. | ");
329                 break;
330             case 5:
331                 printf("Flush | ");
332                 break;
333             case 4:
334                 printf("Straight | ");
335                 break;
336             case 3:
337                 printf("3 cards | ");
338                 break;
339             case 2:
340                 printf("2 pair | ");
341                 break;
342             case 1:
343                 printf("1 pair | ");
344                 break;
345             case 0:
346                 printf("no pair | ");
347                 break;
348 }
349     printf("%4d %3d %3d    %4d %3d %3d\n",
350           score[1][i][0], score[1][i][1],
351           score[1][i][2], score[2][i][0], score[2][i][1], score[2][i][2]);
```

```

351     }
352     printf("\n");
353 }
354
355 // カード管理配列へ、プレーヤーの持っているカードを書き出す
356 int set_card(char mycard, int player) {
357     int suits; // 三つ葉、ダイヤ、ハート、スペード
358     int rank; // トランプに書かれている数字
359
360     suits = ((mycard & 0xf0) >> 4) - 4;
361     rank = mycard & 0x0f;
362
363     // 例 mycard = 'A' の場合
364     // suits = ((mycard & 0xf0) >> 4) - 4;
365     // 'A' = 0x41 = 0b0100_0001;
366     // 'A' & 0xf0
367     // 0x41 & 0xf0
368     // 0b0100_0001
369     // &) 0b1111_0000
370     // -----
371     // 0b0100_0000
372
373     // 0b0100_0000 >> 4 // 右へ 4 ビットシフト
374     // 0b0000_0100
375     // 0b0000_0100 - 4 = 0; // suits = 0となる
376
377     // mycard & 0x0f;
378     // 'A' = 0x41 = 0b0100_0001;
379     // 'A' & 0x0f
380     // 0x41 & 0x0f
381     // 0b0100_0001
382     // &) 0b0000_1111
383     // -----
384     // 0b0000_0001 // 下 4 ビットを取り出すことが出来た
385     // 0b0000_0001 = 1 なので rank(トランプの数字)は 1 と判明する
386
387     if (cards[suits][rank] == 0) {
388         cards[suits][rank] = player;
389         if (rank == 1) {
390             cards[suits][14] = player; // エースの時
391         }
392         return 0; // 正常終了
393     }
394     return cards[suits][rank];
395     // どのプレーヤーで用いられているか返す
396 }
397
398 // 役の計算
399 void calc_card() {
400     int i, j;
401     int sum;
402     int player;
403     int mul;

```

```
404 int delta;
405
406 // フラッシュ ?
407 for (player = 1; player <= 2; player++) {
408     for (j = 0; j <= 3; j++) {
409         sum = 0;
410         for (i = 1; i <= 13; i++) {
411             if (cards[j][i] == player) {
412                 sum += (cards[j][i] / player);
413             }
414         }
415         cards[j][14 + player] = sum;
416     }
417 }
418
419 for (player = 1; player <= 2; player++) {
420     for (j = 0; j <= 3; j++) {
421         if (cards[j][14 + player] == 5) {
422             for (i = 14; i >= 5; i--) {
423                 if (cards[j][i] == player) {
424                     cards[j][14 + player] = i;
425                     break;
426                 }
427             }
428         }
429     }
430 }
431
432 // ペア ?
433 for (player = 1; player <= 2; player++) {
434     for (j = 1; j <= 14; j++) {
435         sum = 0;
436         for (i = 0; i <= 3; i++) {
437             if (cards[i][j] == player) {
438                 sum += (cards[i][j] / player);
439             }
440         }
441         cards[3 + player][j] = sum;
442     }
443 }
444
445 // ストレート ?
446 for (player = 1; player <= 2; player++) {
447     for (delta = 0; delta <= 9; delta++) {
448         mul = 1;
449         for (i = 1 + delta; i <= 5 + delta; i++) {
450             mul *= cards[3 + player][i];
451         }
452         if (mul != 0) {
453             cards[3 + player][14 + player] = --i;
454             break;
455         }
456     }
```

```
457     }
458 }
459
460 // 役の判定
461 void poker_hands(int player) {
462     int work; // 作業用変数
463     int suit;
464     int i;
465
466     work = is_four_of_a_kind(player);
467     suit = is_suit(player, work);
468     score[player][7][1] = work;
469     score[player][7][2] = suit;
470
471     work = is_flush(player);
472     suit = work % 10;
473     score[player][5][1] = work / 10;
474     score[player][5][2] = suit;
475
476     work = is_straight(player);
477     suit = is_suit(player, work);
478     score[player][4][1] = work;
479     score[player][4][2] = suit;
480
481     work = is_three_of_a_kind(player);
482     suit = is_suit(player, work);
483     score[player][3][1] = work;
484     score[player][3][2] = suit;
485
486     work = is_two_pair(player);
487     suit = is_suit(player, work);
488     score[player][2][1] = work;
489     score[player][2][2] = suit;
490
491     work = is_one_pair(player, 14);
492     suit = is_suit(player, work);
493     score[player][1][1] = work;
494     score[player][1][2] = suit;
495
496     work = is_no_pair(player);
497     suit = is_suit(player, work);
498     score[player][0][1] = work;
499     score[player][0][2] = suit;
500
501 // フルハウス ?
502 if (score[player][3][1] != 0 && score[player][1][1] != 0) {
503     score[player][6][1] = score[player][3][1];
504     score[player][6][2] = score[player][3][2];
505 }
506
507 // ストレートフラッシュ ?
508 if (score[player][4][1] != 0 && score[player][5][1] != 0) {
509     score[player][8][1] = score[player][5][1];
```

```
510     score[player][8][2] = score[player][5][2];
511 }
512
513 for (i = 8; i >= 0; i--) {
514     score[player][i][0] = score[player][i][1] * 10 + score[player][i][2];
515 }
516
517 for (i = 8; i >= 0; i--) {
518     if (score[player][i][0] != 0) {
519         score[player][9][0] = score[player][i][0] + i * 1000;
520         score[player][9][1] = score[player][i][1];
521         score[player][9][2] = score[player][i][2];
522         break;
523     }
524 }
525 }
526
527 // フラッシュか判定
528 int is_flush(int player) {
529     int i;
530     for (i = 0; i <= 3; i++) {
531         if (cards[i][14 + player] >= 5) {
532             return cards[i][14 + player] * 10 + i + 1;
533         }
534     }
535     return 0; // フラッシュではなかった
536 }
537
538 // ストレートフラッシュか判定
539 int is_straight(int player) { return cards[3 + player][14 + player]; }
540
541 // フォーカードか判定
542 int is_four_of_a_kind(int player) {
543     int i;
544     for (i = 2; i <= 14; i++) {
545         if (cards[3 + player][i] == 4) {
546             return i;
547         }
548     }
549     return 0;
550 }
551
552 // スリーカードか判定
553 int is_three_of_a_kind(int player) {
554     int i;
555     for (i = 2; i <= 14; i++) {
556         if (cards[3 + player][i] == 3) {
557             return i;
558         }
559     }
560     return 0;
561 }
```

```

563 // ツーペアか判定
564 int is_two_pair(int player) {
565     int work;
566     if ((work = is_one_pair(player, 14)) == 0) {
567         return 0;
568     }
569     if ((is_one_pair(player, work)) == 0) {
570         return 0;
571     }
572     return work;
573 }
574
575 // 10のペアなら10を返す
576 // ペアが見つからなければ0を返す
577 int is_one_pair(int player, int search) {
578     int i;
579     for (i = search; i >= 2; i--) {
580         if (cards[3 + player][i] == 2) {
581             return i;
582         }
583     }
584     return 0;
585 }
586
587 // ノーペアか判定
588 int is_no_pair(int player) {
589     int i;
590     for (i = 14; i >= 2; i--) {
591         if (cards[3 + player][i] == 1) {
592             return i;
593         }
594     }
595     return 0; // error
596 }
597
598 // rank == 13の時、13は何のマークかを返す
599 int is_suit(int player, int rank) {
600     int i;
601     for (i = 3; i >= 0; i--) {
602         if (cards[i][rank] == player) {
603             return i + 1;
604         }
605     }
606     return 0; // rankの数のカードは持っていない
607 }
608
609 // アスキーアートでカードを表示
610 void disp_card(char mycard) {
611     int suits; // 三つ葉、ダイヤ、ハート、スペード
612     char rank; // トランプに書かれている数字
613
614     suits = (mycard & 0xf0) >> 4;
615     rank = mycard & 0x0f;

```

```

616 switch (rank) {
617 case 1:
618     rank = 'A';
619     break;
620 case 11:
621     rank = 'J';
622     break;
623 case 12:
624     rank = 'Q';
625     break;
626 case 13:
627     rank = 'K';
628     break;
629 }
630 // printf("mycard: %x suits: %x, rank: %x\n", mycard, suits, rank);
631
632 switch (suits) {
633 case CLUBS:
634     if (2 <= rank && rank <= 10) {
635         printf("      /\n");
636         printf("    ( )\n");
637         printf("   / %2d \\\n", rank);
638         printf("C-----)\n");
639         printf("      ^\n");
640         printf("      /\_\\ \n");
641     } else {
642         printf("      /\n");
643         printf("    ( )\n");
644         printf("   / %c \\\n", rank);
645         printf("C-----)\n");
646         printf("      ^\n");
647         printf("      /\_\\ \n");
648     }
649     break;
650 case DIAMONDS:
651     if (2 <= rank && rank <= 10) {
652         printf("      /\_\\\n");
653         printf("    / \\\n");
654         printf("   / \\\n");
655         printf("   %2d /\n", rank);
656         printf("   / \\\n");
657         printf("   /\_\\ \n");
658     } else {
659         printf("      /\_\\\n");
660         printf("    / \\\n");
661         printf("   / \\\n");
662         printf("   %c /\n", rank);
663         printf("   / \\\n");
664         printf("   /\_\\ \n");
665     }
666     break;
667 case HEARTS:
668     if (2 <= rank && rank <= 10) {

```

```

669     printf(" /\_/\_/\_\n");
670     printf(" |       |\n");
671     printf(" \_   %2d /\n", rank);
672     printf("    \_ / \_\n");
673     printf("     \_ /\n");
674     printf("      \_\n");
675 } else {
676     printf(" /\_/\_/\_\n");
677     printf(" |       |\n");
678     printf(" \_   %c /\n", rank);
679     printf("    \_ / \_\n");
680     printf("     \_ /\n");
681     printf("      \_\n");
682 }
683 break;
684 case SPADES:
685 if (2 <= rank && rank <= 10) {
686     printf(" /\_/\_\n");
687     printf(" |       |\n");
688     printf(" \_   \_\n");
689     printf(" |   %2d |\n", rank);
690     printf(" \_ / \_\n");
691     printf(" / \_\n");
692 } else {
693     printf(" /\_/\_\n");
694     printf(" |       |\n");
695     printf(" \_   \_\n");
696     printf(" |   %c |\n", rank);
697     printf(" \_ / \_\n");
698     printf(" / \_\n");
699 }
700 break;
701 }
702 }
703
704 /* 0-n未満の数を返す */
705 int my_random(int n) { return clock() % n; }
706
707 // カードをランダムに引く
708 char get_card() {
709     int suits; // 三つ葉、ダイヤ、ハート、スペード
710     int rank; // トランプに書かれている数字
711
712     suits = my_random(4);
713     rank = my_random(13) + 1;
714     return ((suits + 4) << 4) + rank;
715 }
716
717 // 一時停止
718 void pause() {
719     printf(" === press return key === ");
720     while (getchar() != '\n')
721         ;

```

```
722 }
723
724 // 使い方表示
725 void usage(char const *argv[]) {
726     printf(" --- 使い方 --- \n");
727     printf("1");
728     printf("%s -f filename\n", argv[0]);
729     printf("指定されたファイルからplayer1, player2の手を読み込みます\n");
730     printf("\n");
731     printf("2");
732     printf("%s -m\n", argv[0]);
733     printf("交互にランダムでカードを引きます\n");
734     printf("\n");
735 }
```

```
1 ****
2 * A234567890JQK
3 * C ABCDEFGHIJKLMNOP
4 * D QRSTUVWXYZ[\]
5 * H abcdefghijklm
6 * Sqrstuvwxyz{|}
7 *
8 * 例 ) ABCTd と書くと
9 * C(クローバー)のA,2,3,
10 * D(ダイヤ)の4、
11 * H(ハート)の4でワンペア
12 ****
13 /* プレーヤー 1 が所有するカード */
14 EJJKLML
15 /* プレーヤー 2 が所有するカード */
16 ejklml
```

付録 A

珠玉の名著のご紹介

プログラミングに関する名著のご紹介です。ご自身の血肉として頂ければ幸いです。 *¹

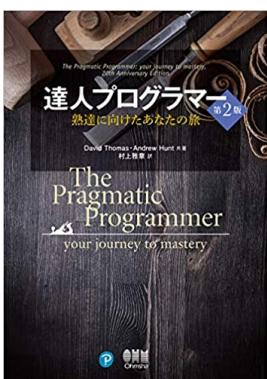
♣ 教養としてのコンピューターサイエンス講義



デジタル時代で活躍するための「教養」をこの1冊で身につけよう。
プリンストン大学の一般人向け「コンピューターサイエンス」の講義が
一冊に。デジタル社会をよりよく生きるために知識を伝説の計算機科学者
がやさしく伝えます。(著者ブライアン・カーニaghan氏は、C言語の發
明者です)

本書は、わたくしたちの世界(デジタル社会)が、どのように動いてい
るのか、なぜそのしくみになっているのかをもっとも明快かつ簡潔に説
明しています。

♣ 達人プログラマー(第2版) 熟達に向けたあなたの旅



本書は、より効率的、そしてより生産的なプログラマーになりたいと願
うソフトウェア開発者に向けて、アジャイルソフトウェア開発手法の先
駆者として知られる二人により執筆されました。経験を積み、生産性を
高め、ソフトウェア開発の全体をより良く理解するための、実践的なアプ
ローチが解説されています。先見性と普遍性に富んだ本書は、入門者には
手引きとなり、ベテランでも読み直すたびに得るものがある、座右の一
冊です。

*¹ 書籍紹介文から、引用・改変。

♣ コーディングを支える技術



本書は、プログラミング言語が持つ各種概念が「なぜ」存在するのかを解説する書籍です。世の中にはたくさんのプログラミング言語があります。そしてプログラミングに関する概念も、関数、型、スコープ、クラス、継承など、さまざまなものがあります。多くの言語で共通して使われる概念もあれば、一部の言語でしか使われない概念もあります。これらの概念は、なぜ生まれたのでしょうか。本書のテーマは、その「なぜ」を理解することです。

そのために本書では、言語設計者の視点に立ち、複数の言語を比較し、そして言語がどう変化してきたのかを解説します。いろいろな概念が「なぜ」生まれたのかを理解することで、なぜ使うべきか、いつ使うべきか、どう使うべきかを判断できるようになるでしょう。

♣ みんなのコンピュータサイエンス



コンピュータなしには生活が立ち行かなくなる水準に達しつつある現代社会。その圧倒的な力を課題解決に援用するには小手先の知識では追いつきません。とは言え無闇に全方位に知識を求めるには、その世界は広すぎ、効率も悪すぎます。

本書は計算機科学が扱う「基礎」「効率」「戦略」「データ」「アルゴリズム」「データベース」「コンピュータ」「プログラミング」という8つのジャンルにしぼり、その精髄と背景となる考え方を紹介します。

ステップアップしたいエンジニアや、ライトに全体像を俯瞰したい学生にも最適な1冊です。

♣ プログラマの数学



プログラミングに役立つ「数学的な考え方」を身につけよう。

プログラミングや数学に関心のある読者を対象に、プログラミング上達に役立つ「数学の考え方」をわかりやすく解説しています。数学的な知識を前提とせず、たくさんの図とパズルを通して、平易な文章で解き明かしています。

二進数から人工知能に至るまで、ていねいに説明しています。

プログラミングや数学に関心のある読者はいうまでもなく、プログラミング初心者や数学の苦手な人にとっても最良の一冊です。

♣ C 言語による標準アルゴリズム事典



コンピュータの算法に関わるアルゴリズムの定石、レトリックを可能な限り収録した定番の書。手元に置いておきたい実用的な本が30年弱の時を経て新装改訂版として登場です。定評をいただいている基本的な内容はそのままに、時代にそぐわなくなっていた部分を改訂。これからも末長くご愛顧いただけるようにまとめ直しました。

♣ アルゴリズム図鑑 絵で見てわかる26のアルゴリズム



基本的な26のアルゴリズム+7つのデータ構造をすべてイラストで解説。アルゴリズムはどんな言語でプログラムを書くにしても不可欠ですが、現場で教わることはめったになく、かといって自分で学ぶには難しいものです。

本書は、アルゴリズムを独学する人のために作りました。はじめて学ぶときにはイメージしやすく、復習するときには思い出しやすくなるよう、基本的な26のアルゴリズム+7つのデータ構造をすべてイラストで解説しています。

よいプログラムを書くために知っておかなければいけないアルゴリズムの世界を、楽しく学びましょう。

♣ C の絵本—C 言語が好きになる9つの扉



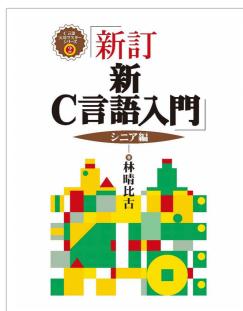
見る見るわかる！【本書の特徴】C言語には難解なトピックもあるため、文章だけではなかなかイメージがつかめず、理解しづらいものですね。本書はイラストで解説してありますので、直感的イメージがとらえられ、理解も進んでいきます。さあ、C言語への扉を開き、プログラマーへの道を進んでいきましょう！

♣ アルゴリズムの絵本-プログラミングが好きになる9つの扉



本書は、プログラミング1年生の方に向けて、プログラムを作る際のアプローチの仕方と初步的なアルゴリズムについて解説した入門書。「プログラムをいかにして組み立てて思い通りに動かすか」を重点的に解説している。特に、頭に浮かんだモヤモヤしたものをプログラムに直す際のアイデアや、ちょっと大きくて複雑なプログラムを作るとときの取り組み方について、イメージをふんだんに使って丁寧に解説している。

♣ 新・C 言語入門-シニア編-C 言語実用マスターシリーズ



本格的なプログラムへの最短コース本格的なプログラム作成に欠かせないC言語の仕様をわかりやすく解説。文法やプログラミングルールの体系的な知識の習得によって、「C言語の思想」も理解できる全プログラマー必読の1冊。困ったときのリファレンスとしても長く活用できる。

♣ C 言語ポインタ完全制覇



C言語で「難しくてよくわからない！」とつまずく人続出るのがポインタ。「Cのポインタがわからないのは、あなたが悪いわけじゃなく、単に、Cの文法がクソなだけだよ!!」第一線で活躍する筆者がCの宣言まわりの混乱した奇っ怪な文法を解き明かし、真のポインタの使い方を教授します。ポインタのみならずCへの理解が一層深まる一冊です。

♣ 小一時間でゲームをつくる 7つの定番ゲームのプログラミングを体験



書いて即実行! シンプルに積み上げていくわくわく感。C言語によるコンソールアプリで、ゲームを一手ずつ作成する手順を解説する、画期的な本の登場です。0から完成まで、手順通りに進めれば必ず完成する仕組みになっています。最小限の手順ごとに動作確認を行うので、それぞれの処理の意味を実感しながら、少しずつできあがっていく過程を楽しめます。

さあ、ゲームプログラミングの旅に出て、難しいクエストも1つずつクリアしていき、夢と冒險に溢れた未知の世界を征服していきましょう!

♣ リーダブルコード より良いコードを書くためのシンプルで実践的なテクニック



コードは理解しやすくなければならない。本書はこの原則を日々のコーディングの様々な場面に当てはめる方法を紹介する。名前の付け方、コメントの書き方など表面上の改善について。コードを動かすための制御フロー、論理式、変数などループとロジックについて。またコードを再構成するための方法。さらにテストの書き方などについて、楽しいイラストと共に説明する。日本語版ではRubyやgroongaのコミッタとしても著名な須藤功平氏による解説を収録。

付録 B

C言語 簡易まとめ

C言語の文法に関する簡易なまとめです。

♣ コメント

プログラミング言語では、ソースコード中に記述されるがコードとしては解釈されない、人に向けた文字列をコメントといいます。主にコードの記述者が別の開発者などにコードの意味や動作、使い方、注意点等について注釈や説明を加える為に使われます。^{*1}

C言語では、コメントは以下のように記述します。

記述例	説明
<code>/* コメント */</code>	複数行コメント
<code>// コメント</code>	一行コメント（便利なので多用されます）

♣ データ型

変数とは、コンピュータプログラムのソースコードなどで、データを一時的に記憶しておくための領域に固有の名前を付けたもの。^{*2}

C言語では、文字型の変数を宣言する際には、`char c;`、整数型の変数を宣言する際には、`int n;`などの型が用意されています。

コード例	説明
<code>char</code>	文字型。 一文字分のアルファベット (1 バイト 2^8 -128～+127までの整数)
<code>int</code>	整数型。 4 バイト 2^{32} -2147483648～+2147483647までの整数)
<code>long</code>	整数型。 8 バイト 2^{64} -9223372036854775808～+9223372036854775807までの整数)
<code>double</code>	倍精度浮動小数点型。 8 バイト 1.7E ± 308(有効 15 衔)までの浮動小数点数)

♣ リテラル

^{*1} 出典：IT 用語辞典

リテラル (literal) とは、直値、直定数とも呼ばれ、コンピュータプログラムのソースコードなどの中に、特定のデータ型の値を直に記載したものである。また、そのように値をコードに書き入れるために定められている書式のことをいう。^{*3}

コード例	説明
123	10進数の整数リテラル
0x30A2	16進数の整数リテラル

♣ 文字列

文字列とは、文字を並べたもの。コンピュータ上では、数値など他の形式のデータと区別して、文字の並びを表すデータを文字列という。^{*4}

C言語には、「文字列」型は用意されていないため、「文字」の「配列」として表します。

コード例	説明
char princess[20] = "shirayukihime";	ダブルクオートの文字列リテラル。
printf("%c", princess[0]);	「s」が出力される。
printf("%s", princess);	「shirayukihime」が出力される。

♣ 演算子

演算子とは、数学やプログラミングなどで式を記述する際に用いる、演算内容を表す記号などのこと。様々な演算子が定義されており、これを組み合わせて式や命令文を構成する。^{*5}

以下の表は優先順位の最も高いものから最も低いものの順に並べられている。^{*6}

*6 出典: 演算子優先順位と結合順序 (<https://www.ibm.com/docs/ja/i/7.5?topic=operators-operator-precedence-associativity>)

コード例	説明
. または->	メンバー選択
[]	添え字
()	関数呼び出し
++	後置増分
--	後置減分
sizeof	サイズ
++	前置増分
--	前置減分
~	ビット単位否定 (1 の補数)
!	否定
-	単項減算
+	単項正
&	アドレス取得
*	間接参照
()	型変換 (キャスト)
*	乗算
/	除算
%	剰余
+	二項加算
-	二項減算
<<	左シフト
>>	右シフト
<	小なり
<=	以下
>	大なり
>=	以上
==	等価
!=	不等価
&	ビット論理積
^	ビット排他的論理和
	ビット論理和
&&	論理積
	論理和
? :	条件式・三項演算子
=	単純代入
*=	乗算代入
/=	除算代入
%=	剰余代入
+=	加算代入
-=	減算代入
<<=	左シフト代入
>>=	右シフト代入
&=	ビット積代入
=	ビット和代入
,	カンマ

♣ 制御構造

プログラムの流れを制御するための構文です。繰り返しのための「**for 文**」、条件分岐のための「**if 文**」などが用意されています。

例	説明
<code>while(x){}</code>	while ループ。 x が true なら反復処理を行う。 繰り返し回数が不明な際に用いると効果的
<code>for(x=0;x < y ;x++){}</code>	for ループ。 x < y が true なら反復処理を行う。 繰り返し回数が分かる時に使うと効果的
<code>if(x){/*A*/*}else{/*B*/*}</code>	条件式。 x が true なら A の処理を、 それ以外なら B の処理を行う
<code>switch(x){case "A":{/*A*/*} "B":{/*B*/*}}</code>	switch 文。 "A" なら A の処理を、 "B" なら B の処理を行う
<code>x ? A: B</code>	条件（三項）演算子。 x が true なら A の処理を、 それ以外なら B の処理を行う
<code>break</code>	break 文。 現在の反復処理を終了しループから抜け出す。
<code>continue</code>	continue 文。 現在の反復処理を終了し次のループに行く。

♣ データアクセス

プログラミング言語 Pascal の開発者 ニクラウス・ヴィルト氏による、「プログラミング」 = 「データ構造」 + 「アルゴリズム」 は、広く知られています。

配列という主要なデータ構造にアクセスするために、次の構文が用意されています。

コード例	説明
<code>array[0]</code>	配列へのインデックスアクセス

♣ 関数宣言

関数とは、コンピュータプログラム上で定義されるサブルーチンの一種で、数学の関数のように与えられた値（引数）を元に何らかの計算や処理を行い、結果を呼び出し元に返すもののこと。

*7

サンプル	説明
<code>int add(x, y){ return x + y; }</code>	関数の一例 仮引数 x と y の和を返す関数

【コラム】金の延棒クイズ 【解答】

最後までお読みくださり、ありがとうございます。金の延棒クイズの解答です。

2回鉢を入れて、金の延棒を1と2と4の大きさに分割します。

一日目のお支払いには、1の延棒を渡します。

二日目のお支払いには、2の延棒を渡して、先に渡した1の延棒は返してもらいます。

三日目のお支払いには、1の延棒も渡します。

四日目のお支払いには、大きな4の延棒を渡し、2と1の延棒は返してもらいます。

五日目のお支払いには、1の延棒も渡します。

六日目のお支払いには、2の延棒を渡して、先に渡した1の延棒は返してもらいます。

七日目のお支払いには、全ての延棒を渡します。

延棒の有無を0と1で表すと二進数と対応しています。

意外なところに潜む二進数。探してみてくださいね。

金の延棒	日当
421	
001	1
010	2
011	3
100	4
101	5
110	6
111	7

終わりに

本書では、算盤から iPhone に至るまでの歴史を俯瞰、計算機科学の基礎知識に触れ、HTML / CSS / JavaScript によるウェブアプリを作成、公開いたしました。

全部で、108行のじんけんプログラム、要所要所にコメントも付けていますので、今まで学んできた知識で読解できるはずです。ぜひ、遊んでみてください。自分で作ったプログラムの体験はいかがでしょうか？いろいろ創意工夫して、さまざまなアプリを作っていくうえですね。

「福祉」。「福」「祉^{*8}」どちらも「めぐみ、さいわい」という意味を持ちます。

「熱き心、^{たくま}逞^{かいな}しき腕、冷静な頭脳」

学生時代に言われた言葉ですが、福祉を生きる者は、人としての熱い思い、暖かい心を持ち、その上で、冷静な判断力を以て、力強く行動するのだと。

「工学」の「工」は、「天の^{ことわり}理^り」を、地に下ろす」意味です。

技術の産物としての社会ではなく、世界を^{かがや}耀^{かがや}かせるために技術を用いてください。技術に使われるのではなく、技術を使いこなし、人の道に役立てる人となってください。

令和の御世を生きる皆さんに素晴らしい人生を生き、素晴らしい日本を創ることを願って筆を置きます。

いやさか
彌榮

*8 「祉い」と書いて、「さいわい」と読みます。天からの恵みがその身に止まる意味です。

はじめてのC言語 練習帳

令和四年九月廿三日 ver 3.0.0

著 者 アトリエ未来

発行者 早乙女 遙香

連絡先 contact@atelier-mirai.net

<https://atelier-mirai.net>

© 令和四年 アトリエ未来