プログラミング及び演習
第2回 実行制御・関数・変数
(2017/04/21)

講義担当
情報学研究科知能システム学専攻
情報基盤センター
教授　森　健策

担当教員・TA

- 講義担当教員
  - 情報学研究科知能システム学専攻　森　健策
  - 電子メール kensaku@is.nagoya-u.ac.jp
  - 電話　052-789-5689
  - 居室　IB館南棟 4F 465室
- 演習担当教員
  - 情報学研究科知能システム学専攻　小田昌宏
  - 電子メール moda@is.nagoya-u.ac.jp
  - 電話　052-789-5688
  - 居室　IB館南棟 4F 466室
- 演習担当TA (ティーチングアシスタント)
  - 情報科学研究科メディア科学専攻 舘高基
  - 電子メール ktachi@mori.m.is.nagoya-u.ac.jp
  - 電話　052-789-5688
  - 居室　IB館南棟 4F 466室

授業の構成 (予定)

- 第1回　4/14　計算機システムの説明/演習準備
- 第2回　4/21　プログラミング概要/実行の制御/関数
- 第3回　5/28　プリプロセッサ/型変換
- 第4回　5/12　演算子/配列1
- 第5回　5/19　配列2 /文字列
- 第6回　5/26　ファイル操作
- 第7回　6/2　ポインタ1
- 第8回　6/16　ポインタ2/標準入出力/デバッガ使用法
- 第9回　6/30　列挙体/構造体
- 第10回 7/7　構造体2
- 第11回 7/14　プログラミングプロジェクト課題（通称 夏休み課題)の解説 / 大演習
- 第12回 7/15　プログラミングプロジェクト（大規模プログラミング）分割コンパイル / make
- 第13回 7/21　プログラミングプロジェクト（大規模プログラミング）デバッガ/ 大演習
- 第14回 7/28　C++に触れてみる
- 第15回 7/28　プログラミングプロジェクト課題（通称 夏休み課題)

授業の構成

- 第1限目 (8:45 – 10:15)
  - 主に講義
  - 教科書などに掲載されたプログラムを打ち込み・実行することでプログラミングに慣れる
- 第2限目(10:30 – 12:00)
  - 主に演習
  - 与えられた演習問題を解くことで，自らのプログラミングを向上させる
  - 演習時間中に提出する課題と1週間以内の提出する課題の2種類
- 両限において出席調査

単位認定

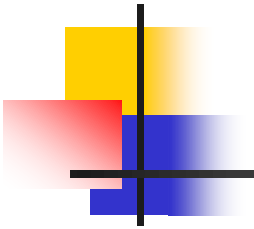
- 達成目標に対しては、以下のように評価する
- 課題レポート70%
- プログラミングプロジェクトレポート 30%
- 100点満点で60点以上を合格とする。
- 補足
  - プログラミングプロジェクトレポート (夏休み課題)は提出必須とする。
  - 課題レポートは演習課題だけではなく、出席調査時の簡単な課題、小テストも含まれる。

講義Webページ・教科書等

- 講義Webページ
  - http://www.newves.org/~mori/17Programming
- 教科書
  - 阿部圭一編, “プログラミング,” オーム社
- 参考書
  - B.W.カーニハン/D.M.リッチー著, 石田晴久訳, “プログラミング言語C 第2版 ANSI規格準拠," 共立出版
  - ハーバート・シルト (著) (柏原 正三 (監修), トップスタジオ (翻訳)), “独習C 第4版,” 翔泳社

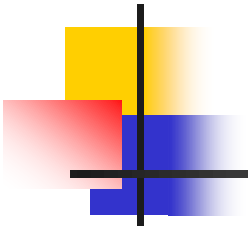
NUCT

- 課題提出・出席チェックで情報メディアセンターのNUCTを利用します。
- 名古屋大学ID/パスワードが必要です。
- 各自NUCT上でコース登録を確認してください。
- 全学メールを通じた通知も行われます
  - 全学メールを定期的にチェックしてください
  - スマホなどに転送すると便利です。
- NUCTはNUPortalと連動しています。



注意

- 「プログラミング及び演習」
  は今年度で最後です。



出席

- NUCTにて出席アンケートに回答する
  - 回答1
    - 今日の合言葉
  - 回答2
    - 「この講義を始めるにあたって一言」を
  - 回答3
    - 「値呼び出しと参照呼出しの違いは分かります？」について、Yes/Noとその違いについて
    - 回答が間違っていても成績には影響しません。

担当教員・TA

- 講義担当教員
  - 情報連携統轄本部情報戦略室/情報基盤センター　森　健策
  - 電子メール kensaku@is.nagoya-u.ac.jp
  - 電話　052-789-5689
  - 居室　IB館南棟 4F 465室
- 演習担当教員
  - 情報科学研究科メディア科学専攻　小田昌宏
  - 電子メール moda@is.nagoya-u.ac.jp
  - 電話　052-789-5688
  - 居室　IB館南棟 4F 466室
- 演習担当TA (ティーチングアシスタント)
  - 情報科学研究科メディア科学専攻 長柄快
  - 電子メール knagara@mori.m.is.nagoya-u.ac.jp
  - 電話　052-789-5688
  - 居室　IB館南棟 4F 466室

授業の構成 (予定)

- 第1回　4/15　プログラミング概要/実行の制御/計算機システムの説明/アンケート
- 第2回　4/22　実行の制御/関数
- 第3回　4/29　プリプロセッサ/型変換
- 第4回　5/6　演算子/配列1
- 第5回　5/13　配列2 /文字列
- 第6回　5/20　ファイル操作
- 第7回　5/27　ポインタ1
- 第8回　6/10　ポインタ2/標準入出力/デバッガ使用法
- 第9回　6/17　列挙体/構造体
- 第10回　7/1　構造体2
- 第11回　7/8　プログラミングプロジェクト課題（通称 夏休み課題)の解説 / 大演習
- 第12回　7/15　プログラミングプロジェクト（大規模プログラミング）分割コンパイル / make
- 第13回　7/16　プログラミングプロジェクト（大規模プログラミング）デバッガ/ 大演習
- 第14回　7/29　C++に触れてみる
- 第15回　7/29　プログラミングプロジェクト課題（通称 夏休み課題)

授業の構成

- 第1限目 (8:45 – 10:15)
  - 主に講義
  - 教科書などに掲載されたプログラムを打ち込み・実行することでプログラミングに慣れる
- 第2限目(10:30 – 12:00)
  - 主に演習
  - 与えられた演習問題を解くことで，自らのプログラミングを向上させる
  - 演習時間中に提出する課題と1週間以内の提出する課題の2種類
- 両限において出席調査

単位認定

- 達成目標に対しては、以下のように評価する
- 課題レポート70%
- プログラミングプロジェクトレポート 30%
- 100点満点で60点以上を合格とする。
- 補足
  - プログラミングプロジェクトレポート (夏休み課題)は提出必須とする。
  - 課題レポートは演習課題だけではなく、出席調査時の簡単な課題、小テストも含まれる。

講義Webページ・教科書等

- 講義Webページ
  - http://www.newves.org/~mori/16Programming
- 教科書
  - 阿部圭一編, “プログラミング,” オーム社
- 参考書
  - B.W.カーニハン/D.M.リッチー著, 石田晴久訳, “プログラミング言語C 第2版 ANSI規格準拠," 共立出版
  - ハーバート・シルト (著) (柏原 正三 (監修), トップスタジオ (翻訳)), “独習C 第4版,” 翔泳社

NUCT

- 課題提出・出席チェックで情報メディアセンターのNUCTを利用します。
- 名古屋大学ID/パスワードが必要です。
- 各自NUCT上でコース登録を確認してください。
  - 2限目に設定を行います。
- 全学メールを通じた通知も行われます
  - 全学メールを定期的にチェックしてください
  - スマホなどに転送すると便利です。
- NUCTはNUPortal (mynu.jp)と連動しています。

計算機環境について

- 本講義ではLinux環境を利用します
- ほぼ同様の環境を仮想マシンイメージファイルとしてWebページを通じて配布します
  - Windowsでも演習室とほぼ同様の環境を再現できます
  - 自宅での演習課題実施、復習、2年後期以降の講義・演習の履修が容易となります
  - 詳しくは次回以降に説明します

Honeywell 6000

POWER REQUIREMENTS

120/208 volts, 3-phase, 4-wire, 60 Hz
Memory (98K): 2.1 KVA, 1.8 KW

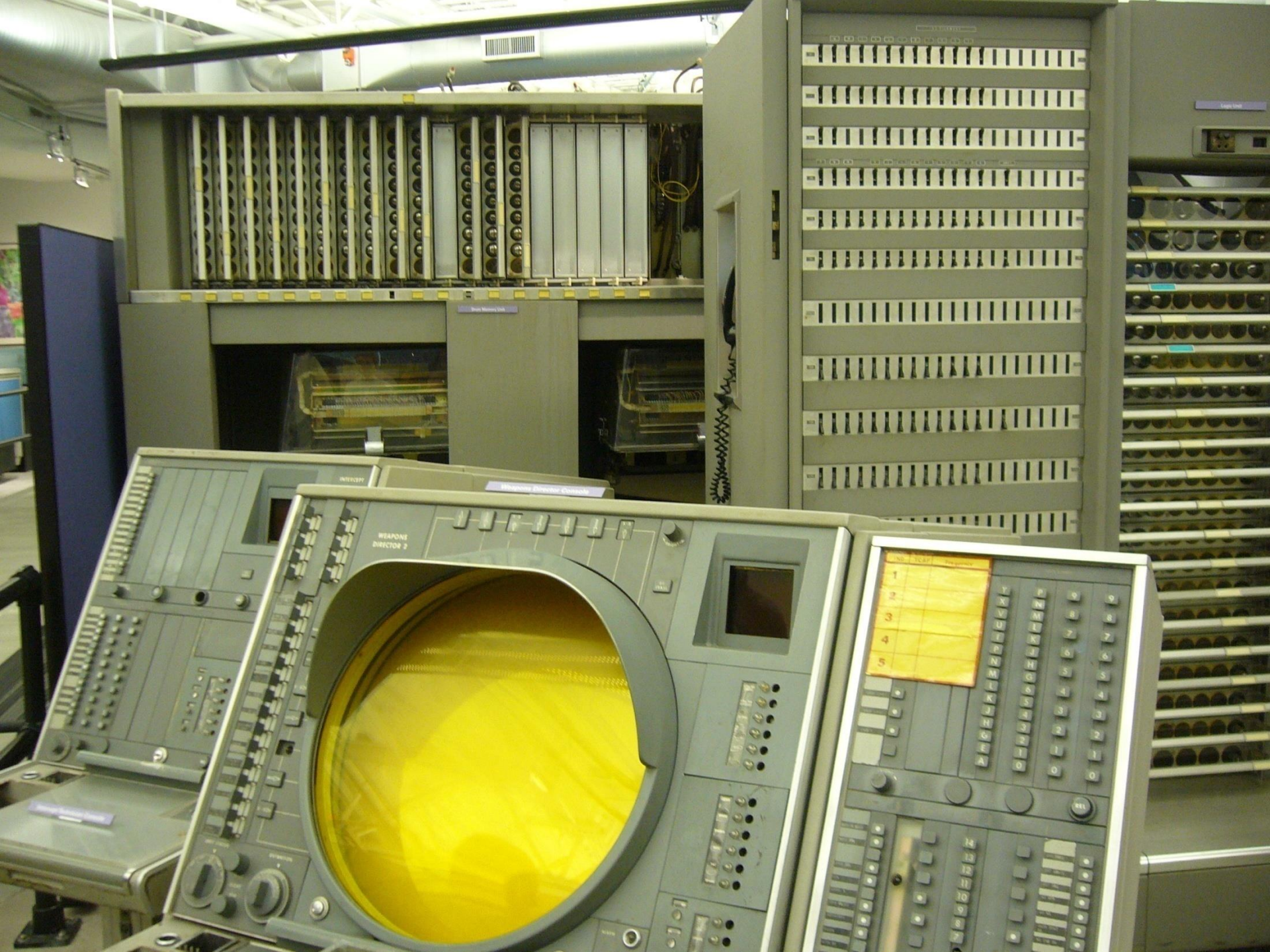
System Controller: 1.2 KVA, 0.8 KW

Processor: 1.4 KVA, 1.3 KW

IOM: 4.0 KVA, 3.6KW

DATANET 355: 3.3 KVA, 2.7 KW

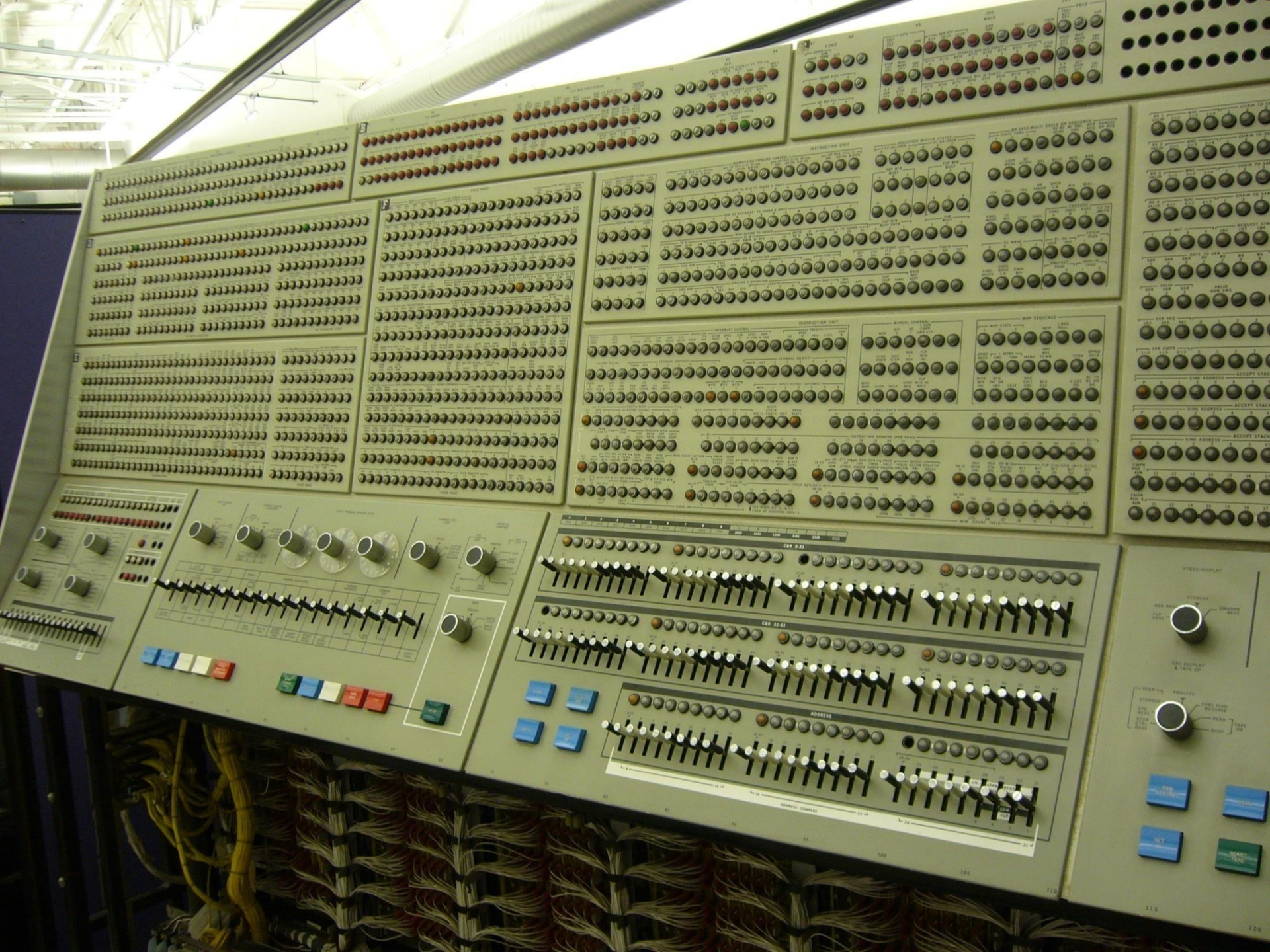
http://home.the-wire.com/~mwilson/his/H6050.html


WEAPONS
DIRECTOR 2


Weapons Director Console
WEAPONS
DIRECTOR 2
PLEASE DO NOT TOUCH
THANK YOU


IBM

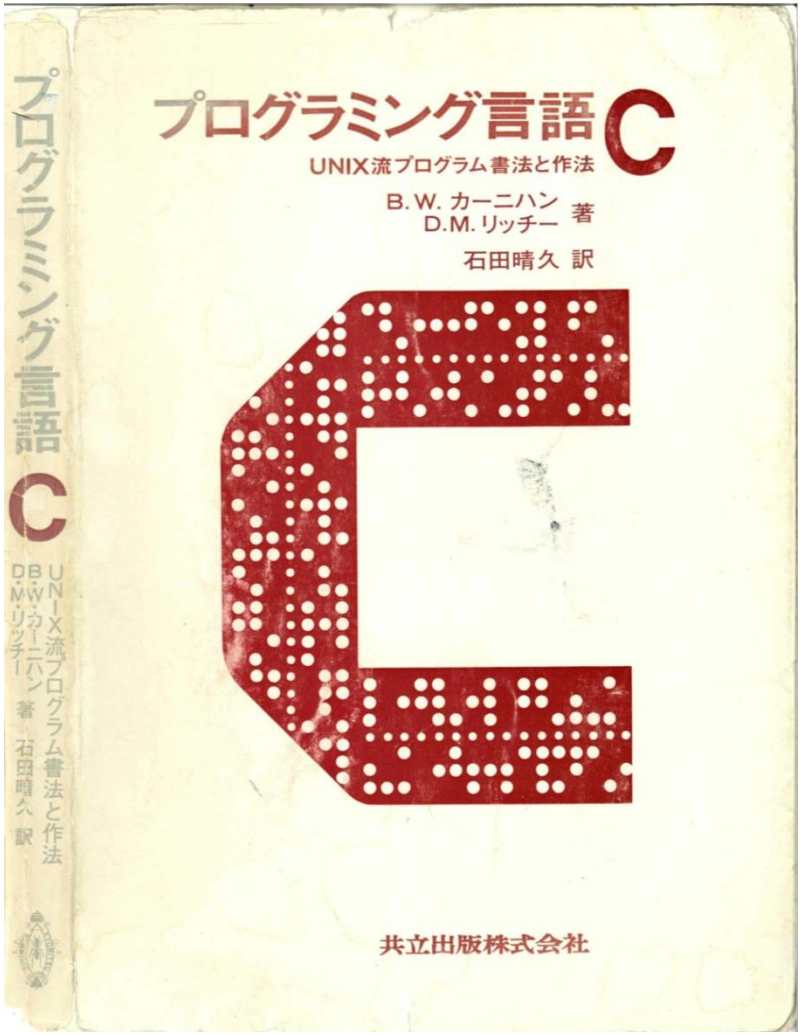

IBM
System 360
ROS CONTROL
RATE
ADDRESS COMPARE
CHECK CONTROL



磁気テープ

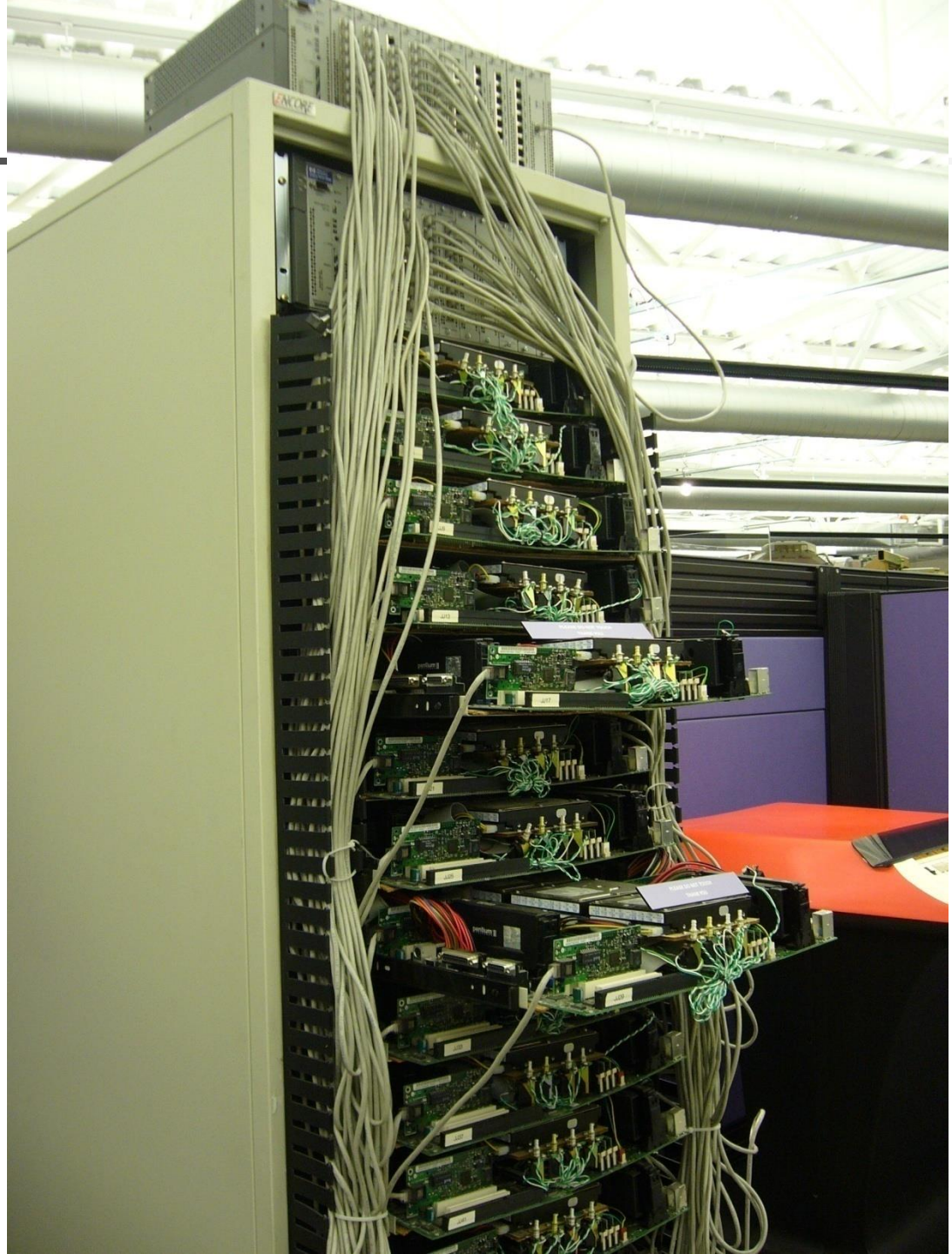

3-D recon data.
1~50 脳幹部ラット.
51~67 ファントムラット.
(High-contrast, Low-contrast)
Scotch
777

プログラミング言語C 第1版



DEC PDP-11

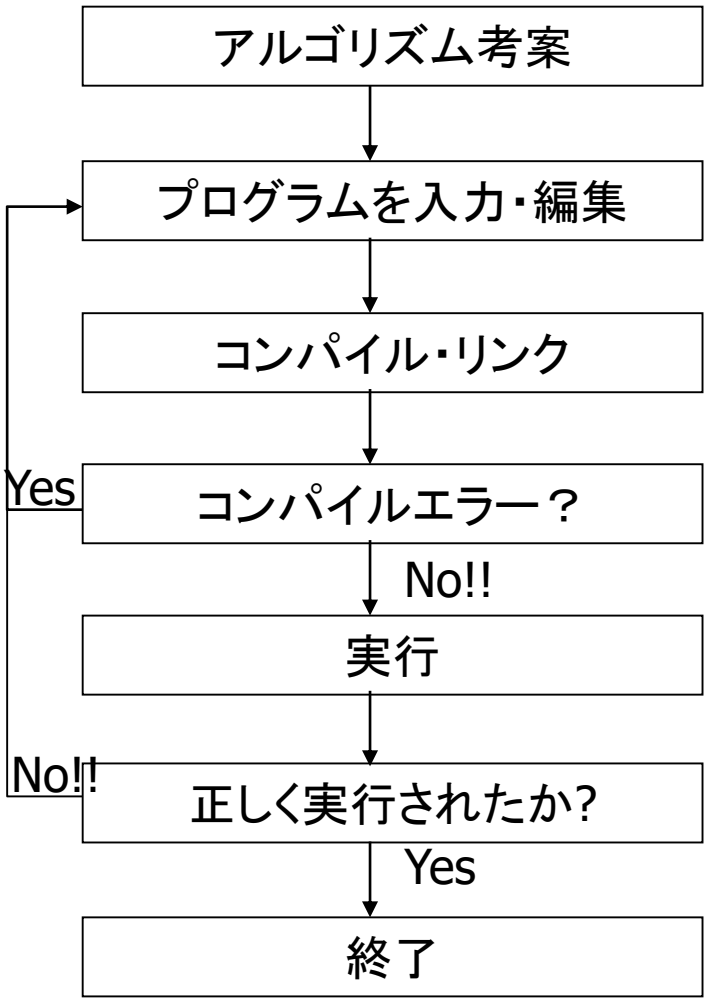

digital
pdp11
ADDRESS REGISTER
DATA
RUN
BUS
FETCH
EXEC
SOURCE
DESTINATION
ADDRESS
Software
DEC PDP-11/20, US, c. 1970
Gift of Tom Mathews (X1507.90)
PLEASE DO NOT TOUCH
THANK YOU
UNIX and the C Programming Language

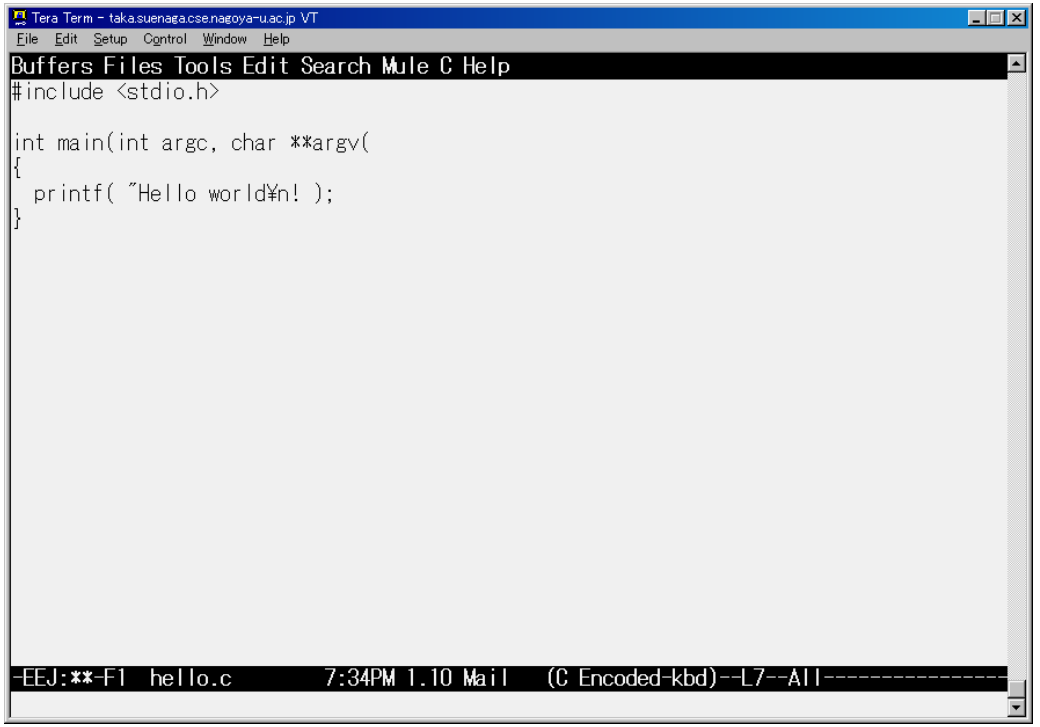


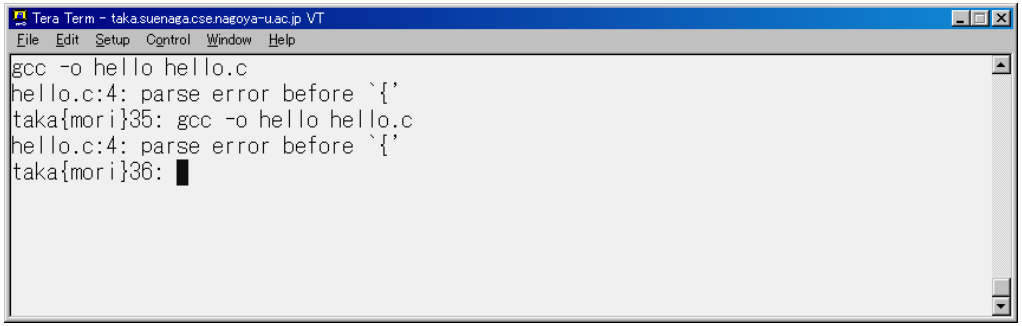
UNIXコマンド基礎中の基礎

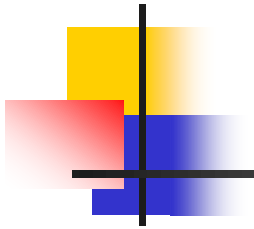
- 必須UNIXコマンド
  - ls - ファイル一覧
  - pwd - 現在のディレクトリを表示
  - cat ファイル名 - ファイルの内容を表示
  - cd ディレクトリ名 - ディレクトリ間の移動
  - rm ファイル名 - ファイルの消去
  - mv ファイル名1 ディレクトリ名 - ファイル移動
  - mv ファイル名1 ファイル名2 - ファイル名変更
  - cp ファイル名1 ディレクトリ名 - ファイルコピー
  - cp ファイル名1 ファイル名2 - ファイルコピー

プログラミングの過程


アルゴリズム考案
プログラムを入力・編集
コンパイル・リンク
コンパイルエラー？
Yes
No!!
実行
正しく実行されたか?
No!!
Yes
終了

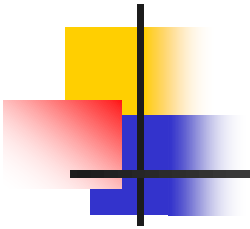

Tera Term - taka.suenaga.cse.nagoya-u.ac.jp VT
File Edit Setup Control Window Help
Buffers Files Tools Edit Search Mule C Help
#include <stdio.h>
int main(int argc, char **argv(
{
printf("Hello world¥n!);
}
-EEJ:**-F1 hello.c 7:34PM 1.10 Mail (C Encoded-kbd)--L7--All


Tera Term - taka.suenaga.cse.nagoya-u.ac.jp VT
File Edit Setup Control Window Help
gcc -o hello hello.c
hello.c:4: parse error before `{'
taka{mori}35: gcc -o hello hello.c
hello.c:4: parse error before `{'
taka{mori}36:



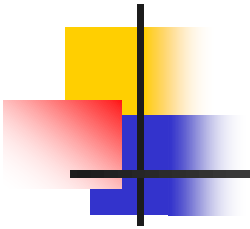
プログラムの復習

```
#include <stdio.h>
main()
{
   printf( “Hello World¥n” );
}
```



変数型

- 基本型　(KR pp.11-12, p.44)
  - char　　文字型
  - int　　整数型　(通常は32bit)
  - float　　単精度浮動小数点型
  - double　　倍精度浮動小数点型
- 修飾子
  - long　　少なくとも32bit
  - short　　少なくとも16bit
  - unsinged　　符号なし
  - singned　　符号あり

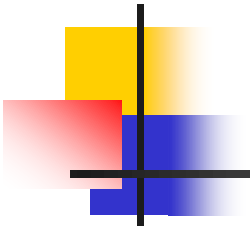


変数の宣言

- Cでは変数を利用する前に必ずその名前を宣言する
  - int num, count;
- 関数の先頭で宣言する
- 変数の値は値を代入するまで不定
- 例
  - int length;
  - float radius=2.5;

変数名の制限

- 最初は必ず英文字 (a-z, A-Z)
- 第2文字以降は数字または下線も利用可
- 演算子等・キーワードは利用不可能
  - “-”, “*”, “+”, “/”, “.” など
  - if, else, for など (KR p.233)
- 正しい変数名の例
  - correct, corrID
- 使えない変数名の例
  - 3floor, 12gatsu, abc+def



変数の利用

- 代入演算子 “=“
  - ＜変数> = <式>;
- 例
  - float num, length;
    length = 3.0;
    num = (12.5+length)*5.2;
  - float x;
    x = 3.0;
    x = x + 3.0;

キーボードから数値の読み込み

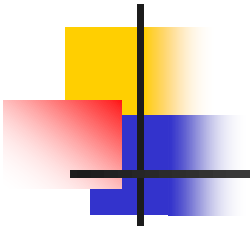
- scanfを利用
  - scanf(“書式”, &<変数>);
  - 書式に従いキーボードから変数に値を取り込む
  - ‘&’はアドレス演算子
  - キーボードからの入力を待ち，書式に従い数字を数値に変換して記憶
  - 入力後には改行キーを入力

サンプルプログラム (教 p.14)

```
#include <stdio.h>
main()
{
        float num;
        printf( “input number: “);
        scanf( “%f”, &num );
        printf( “num*2 is %f¥n”,  num*2.0);
}
```

サンプルプログラム (教 p.15)

```
#include <stdio.h>
main()
{
        int h,m,d;
        printf( "input Hour, Min, Sec: ");
        scanf( "%d, %d, %d", &h, &m, &s);
        printf( "seconds %d", h*3600+m*60+s);
}
```



実行制御

- プログラムは通常上から下に順に実行される
- 実行の制御 (流れの順序を制御)を変更する構文
  - if - else
  - while
  - for
  - do-while
  - switch

条件分岐

- 文法1
  - if (式)
    文1
    else
    文2

式が真ならば文1を実行.
偽ならば文2を実行

- 文法2
  - if (式1)
    文1
    else if (式2)
    文2
    else if (式3)
    文3
    else if (式4)
    文4
    else
    文5

条件分岐文の例 その1

```
if (sec>=60){
    min = min +1;
    sec = 0;
}
```

条件分岐文の例 その2

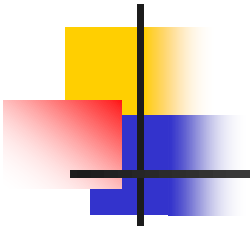
```
if (age>=12){
        fare = 1000;
        adult = 1;
}else{
        fare = 500;
        adult = 0;
}
```

条件分岐文の例 その3

```
if (price==50){
        printf( “price = 50¥n");
}else if (price==100){
        printf( “price = 100¥n");
}else if (price==150){
        printf( “Price = 150¥n");
}else{
        printf( “???¥n");
}
```

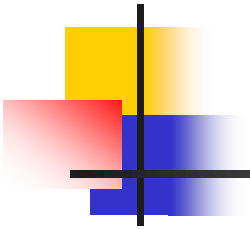
講義中課題2-1 (lec2-if.c)

- 以下の条件分岐構造で {}が無い場合どのようになるか試してみよ(if, elseとも無い場合，else側に無い場合)
- if (age>=12){
  fare = 1000;
  adult = 1;
  }else{
  fare = 500;
  adult = 0;
  }



比較演算

- 比較演算子
  - >, <
  - >=, <=
  - ==, !=
- 比較演算子の使い方
  - <左辺の式><比較演算子><右辺の式>

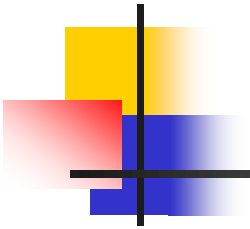


論理演算

- 論理演算子
  - ||
  - &&
  - !
- 例
  - (count>0) && (count<10)
  - (a<b) || (c==d)
  - !(a>b) && (c==d)

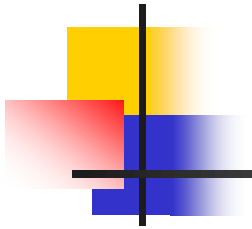
繰り返し処理 (1) while

- 文法
  - while(<式>)
    <文>
- <式>が真である間<文>を繰り返し実行する
- <文>が一度も実行されない場合もある



whileの例 その1

```
int count = 0;
while (count<10){
        printf( "count %d¥n", count );
        count = count +1;
}
printf( "count %d¥n", count );
```



whileの例 その2

```
#include <stdio.h>
main()
{
    int count=0, sum=0;
    while(count<10){
        sum = sum + count;
        printf( "count %d sum %d¥n", count, sum );
        count = count +1;
    }
}
```

華氏　Fahrenheit

- C = (F-32)*5/9

- (以下Wikipediaから)
  - ～10度台 - 厚い霜が降りる。即座に凍え死ぬ寒さ。
  - 20度台 - 薄い霜が降りる。
  - 30度台 - 寒い。氷点に近い。極寒。
  - 40度台 - 寒い。厚い衣服が必要。
  - 50度台 - 涼しい。適度な厚さの衣服で十分。運動には適温。
  - 60度台 - 暖かい。薄手の衣服が必要。
  - 70度台 - 適度に暑い。夏服が必要。
  - 80度台 - 暑いが耐えられる。少なめの衣服。猛烈な暑気。
  - 90度台 - とても暑い。過熱に対する予防措置が必要。
  - 100度台～ - 危険なほど暑い。生存には危険な酷暑。

Weather Channelより

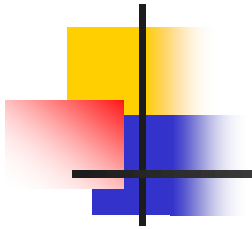
華氏から摂氏への変換

```
#include <stdio.h>
main()
{
  float fahr, celsius;
  int lower, upper, step;
  lower = 0;
  upper = 300;
  step = 20;
  fahr = lower;
  while(fahr<=upper){
    celsius = (5.0/9.0)*(fahr-32.0);
    printf( "%3.0f %6.1f¥n", fahr, celsius);
    fahr = fahr + step;
  }
}
```

```
  0  -17.8
 20   -6.7
 40    4.4
 60   15.6
 80   26.7
100   37.8
120   48.9
140   60.0
160   71.1
180   82.2
200   93.3
220  104.4
240  115.6
260  126.7
280  137.8
300  148.9
```

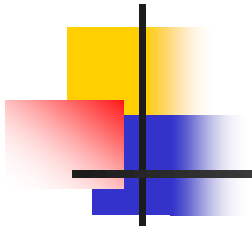
繰り返し処理 (2) for

- 文法
  - for(<式1>; <式2>; <式3>)
    <文>
- <式1>により初期化し、<式2>が真の間<文>を実行する。その後<式3>を実行。



forの例 その1

```
int count;
for(count=0; count<10; count++){
      printf( "count %d¥n", count);
}
```



forの例 その2

```
int count, sum;
sum = 0;
for(count=0; count<10; count++){
    sum = sum + count;
    printf( "count %d sum %d¥n", count, sum);
}
printf( "sum = %d¥n", sum );
```

forの例　その3　多重ループ

```
#include <stdio.h>
main()
{
  int i,j;
  for(j=0; j<3; j++){
    for(i=0;i<4; i++){
        printf( "i=%d j=%d¥n",i,j);
    }
  }
}
```

講義中課題2-2 (lec2-conv.c)

- 右の華氏・摂氏変換プログラムをfor文を使った形に書き換える

```
#include <stdio.h>
main()
{
  float fahr, celsius;
  int lower, upper, step;
  lower = 0;
  upper = 300;
  step = 20;
  fahr = lower;
  while(fahr<=upper){
    celsius = (5.0/9.0)*(fahr-32.0);
    printf( "%3.0f %6.1f¥n", fahr, celsius);
    fahr = fahr + step;
  }
}
```

繰り返し処理 (3) do-while

- 文法
  - do
    <文>
    while(<式>)
- <文>を実行した後、<式>を評価。
  真であるならば<文>を再度実行
- while文とは違い<文>は必ず1回は実行される

do-whileの例

```
#include <stdio.h>
main(){
    int count=0, sum=0;
    do{
        sum = sum + count;
        printf( "count %d sum %d¥n", count, sum );
        count = count +1;
    } while(count<10);
}
```

講義中課題2-3 (lec2-dowhile.c)

- 以下のプログラムをfor, whileを使った形に書き換える

```
#include <stdio.h>
main(){
    int count=0, sum=0;
    do{
        sum = sum + count;
        printf( "count %d sum %d¥n", count, sum );
        count = count +1;
    } while(count<10);
}
```

多分岐判断 switch

- 文法
  - switch(<式>){
    定数1: <文1>;
    <文2>;
    break;
    定数2: <文3>;
    <文4>;
    break;
    default: <文7>;
    <文8>;
    }

<式>を評価した後，その値に従い
定数1であれば<文1>,<文2>を実行，
定数2であれば<文3>,<文4>を実行，
それ以外ならば<文7>,<文8>を実行

breakがないと後続の分も実行されるので注意！！

switchの例

```
taka{mori}18: ./a.out
num:10
case9
num:1
case 1
num:2
case 2
case 3
num:0
case 0
case 1
num:case 0
case 1
num:-1
case9
taka{mori}19:
```

```
#include <stdio.h>
main(){
  int num=0;
  while(num>=0){
    printf( "num:" );
    scanf("%d",&num);
    switch(num){
    case 0:
      printf( "case 0¥n" );
    case 1:
      printf( "case 1¥n" );
      break;
    case 2:
      printf( "case 2¥n" );
    case 3:
      printf( "case 3¥n" );
      break;
    default:
      printf( "case9¥n" );
      break;
    }
  }
}
```

breakとcontinue (KR p.78)

- break
  - ループを抜け出るための命令
  - 例)

```
for(i=0; i<10; i++){
 if(i==5) break;
 printf( "i=%d¥n",i);
}
```

breakとcontinue (KR p.78)

- continue
  - for, do, whileの次の繰り返しを開始
  - 例)

```
for(i=0; i<10; i++){
  if(i==5) continue;
  printf( "i=%d¥n",i);
}
```

講義中課題2-4 (lec2-break.c)

- 以下のプログラムを入力して動作の違いを確かめよ
  - for(i=0; i<10; i++){
    if(i==5) break;
    printf("i=%d¥n",i);
    }
    for(i=0; i<10; i++){
    if(i==5) continue;
    printf("i=%d¥n",i);
    }

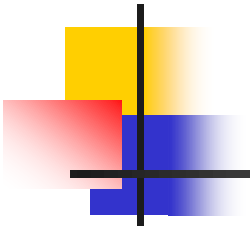
講義中課題2-5 (lec2-forbreak.c)

- 以下のプログラムの動作はどのようになるか

```
#include <stdio.h>
main()
{
  int i,j;
  for(j=0; j<3; j++){
    for(i=0;i<4; i++){
      if (i==2) break;
      printf( "i=%d j=%d¥n",i,j);
    }
  }
}
```

これがスライドです


実験結果
症例No
木構造化された枝数
従来手法で対応づけられた枝数
本手法で対応づけられた枝数
001 28 24 26
002 27 23 25
005 9 9 9
006 9 9 19
007 19 19 7
010 19 15 9
014 9 9 9
018 9 9 9
019 9 9 43
021 43 43 17
024 17 7
対応づけ結果
偽枝
従来手法
本手法
実験試料
画像サイズ 512pixel×512pixel
画素間隔 0.625mm×0.625mm
スライス枚数 50~183
スライスの厚さ 2.0mm or 5.0mm
再構成間隔 1.0mm or 2.0mm
症例数 11
使用計算機
SGI INDIGO2 HighIMPACT (CPU R4400)
平均計算時間
1症例あたり平均11秒
辺縁枝の枝名対応づけ
従来手法と同一の処理
利用する特徴量
自身の枝名
親の枝名
走行方向
主要枝の子枝からの深さ優先探索
枝名対応づけ



関数

- ある一連の手続きを一つの文で呼び出し実行する機能
  - サブルーチン
  - 副プログラム
  - 手続き (procedure)

関数の使用例

```
#include<stdio.h>

int abs(int x)
{
   if(x>=0){
      return(x);
   }else{
      return(-x);
   }
}
```

```
main()
{
    int x,y,z;
    y=-5;  z=7;
    x=abs(y);
   printf( "abs(%d)=%d",
                    y,x);
    x=abs(y+z);
   printf( "abs(%d)=%d",
                    y+z,x);
}
```

関数の呼び出し方法

- <関数>(<引数1>, <引数2>,
  ＜引数3>,……., <引数n>);

関数の定義

```
<関数の型> <関数名>(<型> <引数1>,
                    <型> <引数2>,…,
                    <型> <引数3>)
{
    <文1>;
    <文2>;
    ….
    return <式>;
}
```

- 引数で渡される値を用いて処理を行う
- 結果の値をreturnで返す
- 返す値が無い場合は関数の型をvoidとする

関数の使用例

```
#include <stdio.h>
int power(int b, int n)
{
    int i,p;
    p=1;
    for(i=1;i<=n;i++){
        p=p*b;
    }
    return p;
}
```

```
main()
{
    int i;
    for(i=0;i<=10;i++){
        printf( "%d %d %d¥n",
                power(2,i),
                power(-3,i) );
    }
}
```

関数プロトタイプ (教 p.35-)

- 関数を定義する前に呼び出す場合
  - 関数･引数の宣言をする
  - コンパイル時に呼び出し方のチェックがなされる
- 記述法
  - <関数の型> <関数名>(<型> <引数1>,
    <型> <引数2>,…,
    <型> <引数3>);
  - 関数宣言の部分と全く同じ
  - 引数1-3は省いてもOK

関数の使用例

```
#include <stdio.h>
int power(int b, int n);
main()
{
    int i;
    for(i=0;i<=10;i++){
    printf( "%d %d %d¥n",
        power(2,i),
        power(-3,i) );
    }
}
```

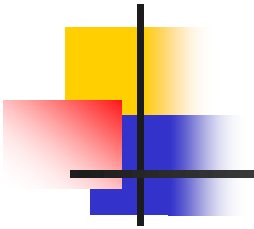
```
int power(int b, int n)
{
    int i,p;
    p=1;
    for(i=1;i<=n;i++){
        p=p*b;
    }
    return p;
}
```

講義内課題 (lec2-power1.c)

以下のプログラムでプロトタイプ宣言がないとどうなるか? また呼び出し型が異なると?

```
#include <stdio.h>
int power(int b, int n);
main()
{
    int i;
    for(i=0;i<=10;i++){
        printf( "%d %d %d¥n",
         power(2,i),
         power(-3,i) );
    }
}
```

```
int power(int b, int n)
{
    int i,p;
    p=1;
    for(i=1;i<=n;i++){
        p=p*b;
    }
    return p;
}
```



Call by value (値による呼び出し) (⇔ Call by referene) (KR p.34-)

- Cではすべての関数の引数が値で呼び出される
  - 呼び出し元の変数ではなく、一時変数に値がコピーされて呼び出される
- 呼ばれた関数は呼んだ関数の値を変えることができない
- プライベートな一時的な変数を変えることは可能

一時変数に値がコピーされる

```
int power(int b, int n)
{
    int i,p;
    p=1;
    for(i=1;i<=n;i++){
        p=p*b;
    }
    return p;
}
```

```
int power(int b, int n)
{
    int p;
    for(p=1;n>0;--n){
        p=p*b;
    }
    return p;
}
```

powerを呼ぶ側のnは呼ぶ前後で変化しない!

講義内課題 (lec2-power2.c)

以下のプログラムで呼ぶ側の変数は関数呼び出しによって変化しないことを確かめよ

```
#include <stdio.h>
int power(int b, int n);
main()
{
    int i,k;
    for(i=0;i<=10;i++){
        k=i;
        printf( "before %d¥n",k);
        printf( "pow %d ¥n",
         power(2,k));
        printf( "before %d¥n",k);
    }
}
```

```
int power(int b, int n)
{
    int p;
    for(p=1;n>0;--n){
        p=p*b;
    }
    return p;
}
```

講義内課題 (lec2-power3.c)

以下のプログラムで呼ぶ側の変数は関数呼び出しによって変化することを確かめよ

```
#include <stdio.h>
int power(int b, int n);
main()
{
  int i,k;
  for(i=0;i<=10;i++){
    k=i;
    printf( "before %d¥n",k);
    printf( "%d¥n",
      power(2,&k));
    printf( "before %d¥n",k);
  }
}
```

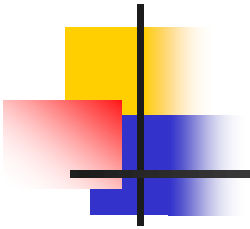
```
int power(int b, int *n)
{
  int p;
  for(p=1;(*n)>0;--(*n)){
      p=p*b;
  }
  return p;
}
```

関数の定義

```
<関数の型> <関数名>(<型> <引数1>,
                    <型> <引数2>,…,
                    <型> <引数3>)
{
    <文1>;
    <文2>;
    ….
    return <式>;
}
```

- 引数で渡される値を用いて処理を行う
- 結果の値をreturnで返す
- 返す値が無い場合は関数の型をvoidとする

Call by value (値による呼び出し) (KR p.34-)



- Cではすべての関数の引数が値で呼び出される
  - 呼び出し元の変数ではなく、一時変数に値がコピーされて呼び出される
- 呼ばれた関数は呼んだ関数の値を変えることができない
- 一時的にプライベートな変数を変えることは可能

一時変数に値がコピーされる

```
int power(int b, int n)
{
    int i,p;
    p=1;
    for(i=1;i<=n;i++){
        p=p*b;
    }
    return p;
}
```

```
int power(int b, int n)
{
    int p;
    for(p=1;n>0;--n){
        p=p*b;
    }
    return p;
}
```

powerを呼ぶ側のnは呼ぶ前後で変化しない!

局所変数 (教p.36-)

- 関数の内部で宣言された変数　＝ 局所変数
  - その関数内部でのみ有効
  - 外部からアクセスすることは不可能
  - 異なる関数中で同じ変数名前を使用可能
  - 仮引数も局所関数
  - 関数が呼び出されるときに現れ、終了後消える
    - 関数の実行が終了すると変数の領域が開放される

大域変数 (教p.36-)

- 関数の外部で宣言された変数＝大域変数
  - すべての関数において有効
  - 広域変数と同じ変数名で局所変数を宣言したときはその関数内では局所変数が有効
  - プログラミングのときは最小限とすることが望ましい

講義内課題 (lec2-var.c)

```
#include <stdio.h>
int x,y;
void fa()
{
    int x,i;
    i=0; x=1; y=2;
}
void fb()
{
    int i;
    i=0; x=1; y=2;
}
```

```
main()
{
    x=10;
    y=20;
    fa();
    printf( "%d %d¥n", x,y );
    fb();
    printf( "%d %d¥n", x,y );
}
```

(関数内での)静的変数 (教 p.46)

- 関数内で保持される変数 = 静的変数
- 宣言時にstaticをつける
  - static int k;
- 関数呼び出し終了後も変数の値が保持される

講義内課題 (lec2-stat.c)

以下のプログラムの動作を確かめよ　cの値はどうなるか?

```
#include <stdio.h>
void count();
main()
{
    int i,k;
    for(i=0;i<=10;i++){
        count();
    }
}
```

```
void count()
{
    static int c=0;
    c++;
    printf("count =%d¥n",c);
}
```

再帰呼び出し (教 p.39)

- 自分自身を関数内で(直接的・間接的)に呼び出すことができる
- 再帰の形で書かれたアルゴリズムを実装するのに便利
- スタックオーバーフローに注意
- フィボナッチ数列
  - fib(0)=1;
    fib(1)=1;
    fib(i)=fib(i-1)+fib(i-2);

講義内課題 (lec2-fib.c)

1. num=100としたとき何回fibが呼ばれるかカウントしてみよ
2. limit stacksize 100k としてから num =100として実行してみよ

```
#include <stdio.h>
int fib(int i)
{
    printf( "enter fib(%d)¥n",i);
    if((i==0)||(i==1)){
        return(1);
    }else{
        return(fib(i-1)+fib(i-2));
    }
}
```

```
int main()
{
    int num;
    printf("num ¥n");
    scanf("%d",&num);
    printf("fib(%d)=%d¥n", num,
        fib(num));
    return 0;
}
```

数字を文字列として印字
講義内課題 (lec2-printd.c)

以下のプログラムの動作を確かめよ. printfはどのように呼び出されるか

```
#include <stdio.h>
void printd(int n)
{
  if(n<0){
    putchar('-');
    n=-n;
  }
  if(n/10){
    printd(n/10);
  }
  putchar(n%10+'0');
}
```

```
main()
{
  int num;
  printf("num = ");
  scanf("%d",&num);
  printd(num);
}
```