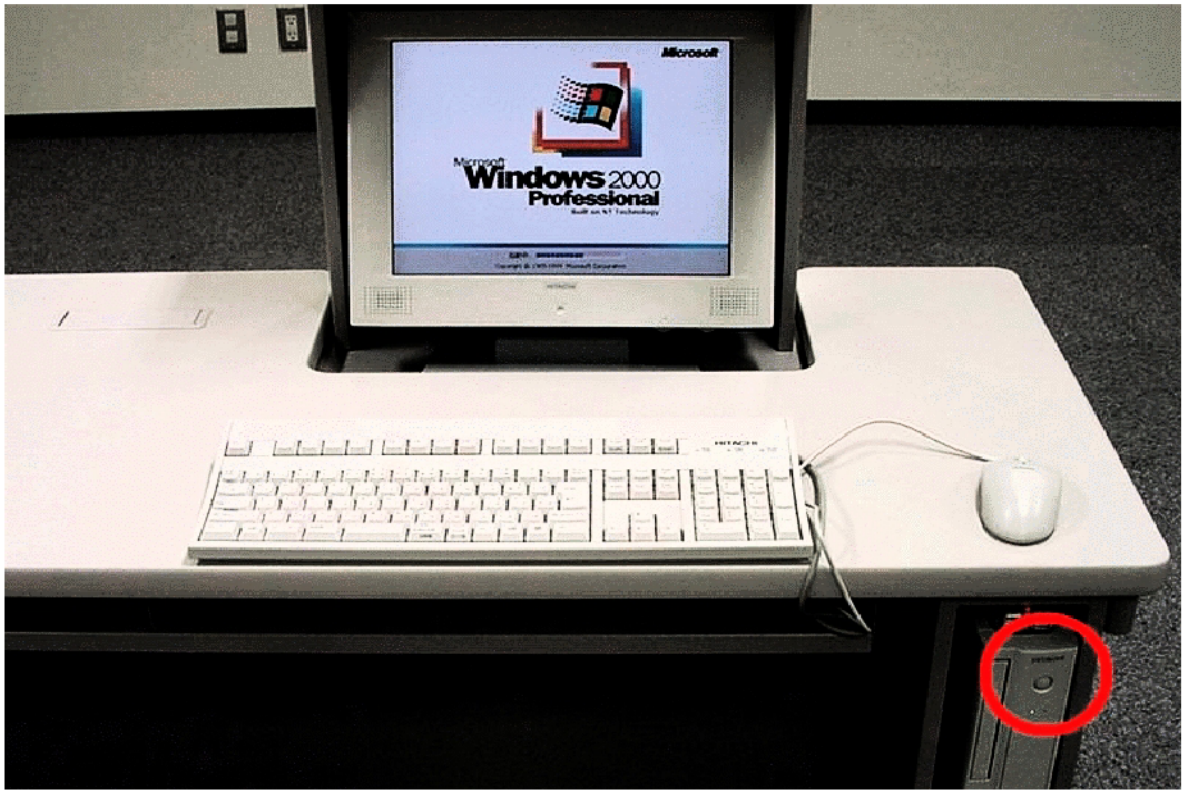
情報処理論2a

0章「プログラムによる問題解決」

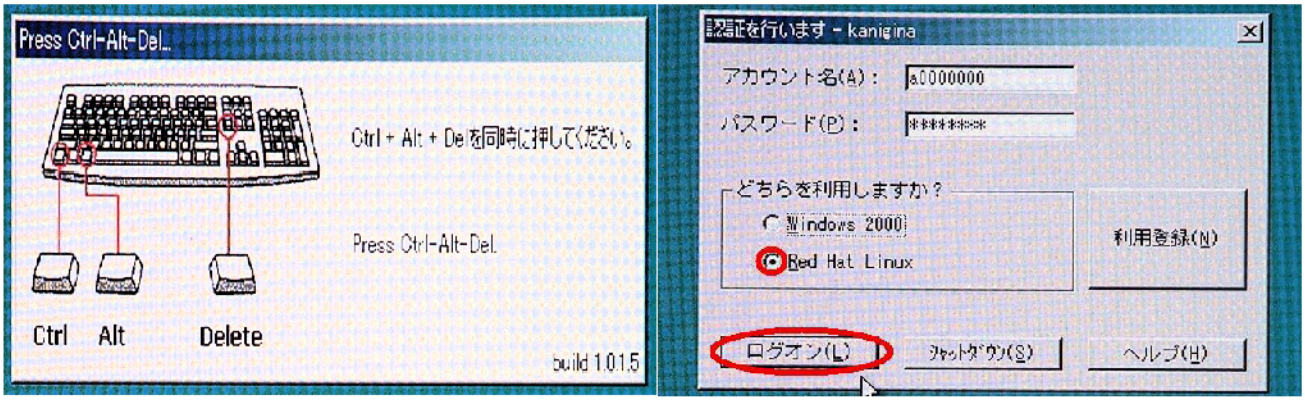
中島康彦

§0. 2 電源の投入



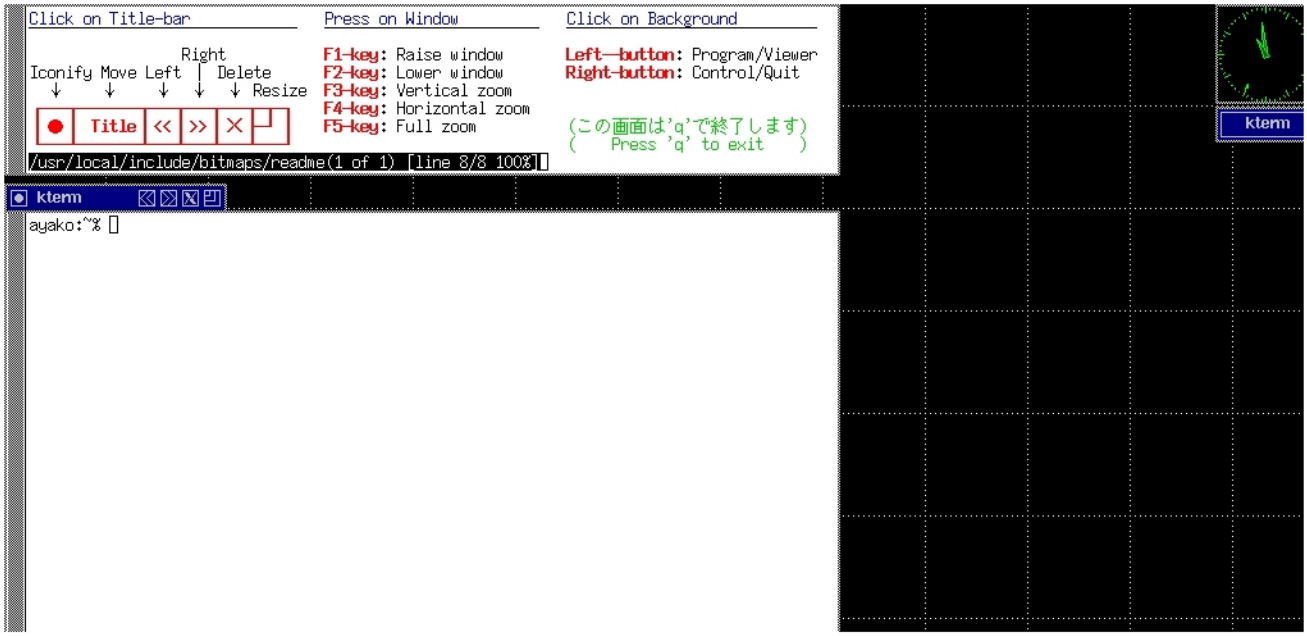
§0. 3 Xwindow

各自の利用コードおよびパスワードを使ってログイン



- 本来Windowsは不要だが，管理の都合上，Windows上で動作する仮想計算機を使っている.
- VMwareが起動するのを待つ.

§0. 3 Xwindow



- Xwindowが起動し，kterm(ターミナルウィンドウ)が開く.
- 組み込み用LINUXがそうであるように，本来GUIは不要.
- ktermの中にマウスカーソルを移動すると文字入力可能.
- exit と入力し，ktermを閉じる.

§0. 4 講義資料の表示

1. ktermを起動
2. % netscape http://www.econ.kyoto-u.ac.jp/~nakashim/
3. 「表示」→「文字コード」→「日本語(自動判別)」を選択
4. 「情報処理論1a」の「1章 UNIXの基本操作とメール」を選択
5. 終了する時は,「ファイル」→「終了」

§0. 5 準備

UNIXを使い易くする設定(一度だけ実行)

1. 1章の演習に必要なファイル(data01)の上で右ドラッグし「リンクを名前を付けて保存」

2. 設定ファイルの初期化

```
% cd                  ⇒ ホームディレクトリに移動
% tar xvf data01⇒ 初期化
% ls -a              ⇒ ファイル一覧の表示
./      ../      .Xdefaults        .cshrc
.emacs20.el   .emacs-color.el
.emacs19.el   .emacs              Bin/
```

3. ktermを閉じる

```
% exit
```

§0. 6 今日の作業ディレクトリを作る

1. % cd ⇒ ホームディレクトリへ移動

2. % mkdir chap12 ⇒ ディレクトリchap12を作成

3. % cd chap12 ⇒ ディレクトリchap12へ移動

4. % pwd ⇒ 作業ディレクトリの確認

§0. 7 ソースプログラムが実行されるまで

1. C言語ソースsample.cを作る.

```
% vi sample.c
main()
{
  printf("Hello.\n");
}
```

2. Cコンパイラを使って実行可能形式を生成してみる.
4つのコマンドが順に呼び出されて順にファイルを生成する.

```
sample.c ⇒ cxxx.i ⇒ cxxx.s ⇒ cxxx.o ⇒ sample
% gcc -v sample.c -o sample
.../cpp ...        sample.c      .../cxxx.i
.../cc1 ...        .../cxxx.i -o  .../cxxx.s
.../as  ...        .../cxxx.s -o  .../cxxx.o
.../ld  ...        .../cxxx.o -o  sample
```

3. 実行

```
% ./sample
Hello.
```

§0. 8 アセンブラ(as)

cxxx.sからオブジェクトcxxx.oを作成する.
アセンブリ言語を解釈しバイナリ形式に変換.

【Pentiumの場合】
```
main:   pushl %ebp
        movl %esp,%ebp
        pushl $.LC0
        call printf
        leave
        ret
.LC0:   .ascii "Hello.\12\0"
```

【SPARCの場合】
```
main:   save %sp,-112,%sp
        sethi %hi(.LLC0),%o0
        call printf,0
        or %o0,%lo(.LLC0),%o0
        ret
        restore
.LLC0:  .asciz  "Hello\n"
```

【HPPAの場合】
```
main    ldil LR'L$C0000,%r26
        bl printf,%r2
        ldo RR'L$C0000(%r26),%r26
        ldw -148(0,%r30),%r2
        bv 0(%r2)
        ldo -128(%r30),%r30
L$C0000 .STRING "Hello.\x0a\x00"
```

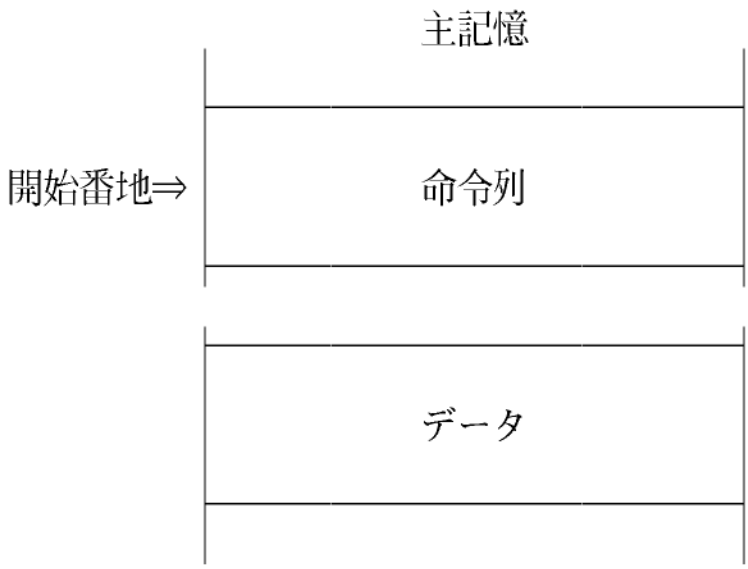
§0. 9 リンカ(ld)

cxxx.oからロードモジュールsampleを作成する.
初期化関数やライブラリを結合する.

ロードモジュール

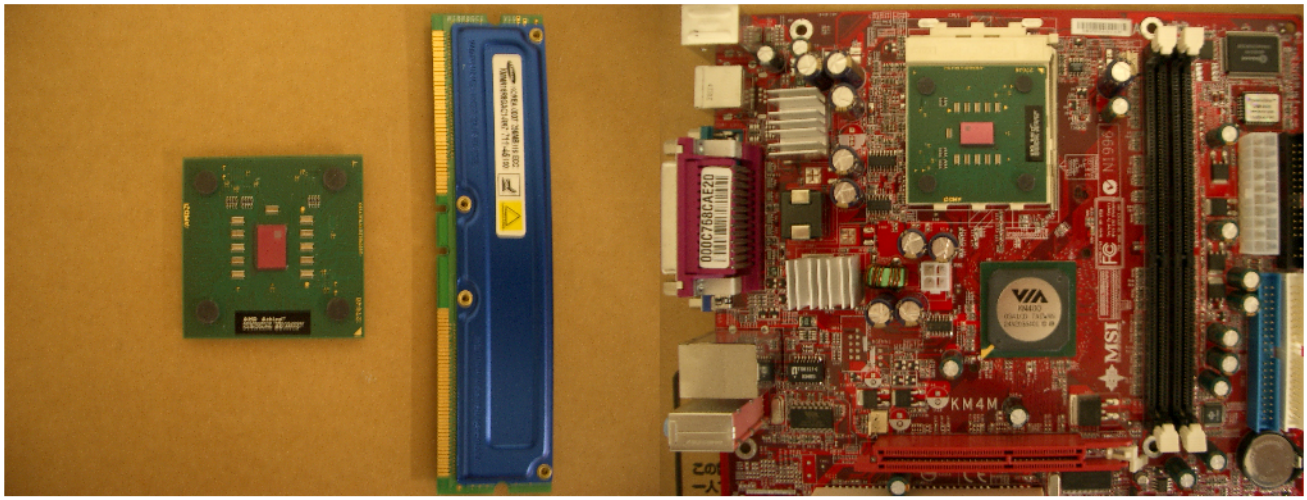
§0. 10 実行

ロードモジュールsampleが主記憶に展開され，ヘッダに書いてある開始番地から命令列が実行される.
主記憶に展開された状態をコアイメージと呼ぶ.



§0. 10 実行

CPU … ￥2万
メモリ … ￥5千
マザーボード … ￥1万
電源 … ￥5千



§0. 11 ソフトウェア開発とは

機能設計(Functional Design)
- 外部仕様書…開発者と利用者のインタフェース

構成設計(Structural Design)
- 内部仕様書…共同開発のための内部インタフェース

詳細設計(Detailed Design)
- ソースプログラム

論理試験(Logical Test)
- 入力データと出力期待値

耐性試験(Error Tolerance Test)
- 異常に対する耐性/検出能力

§0. 12 例題

sample.c ⇒ cxxx.i ⇒ cxxx.s ⇒ cxxx.o ⇒ sample

ソースプログラムから作成される上記ファイルのうち，CPUの差異を気にせずに，異なるシステムでも再利用可能なファイルはどれであるか，また，その理由を解答せよ．(エミュレータなどの特殊な機構は用いない)

§0. 13 今日の課題

CPUの差異に関連して，以下のコンピュータでは，どのようなCPUとOS(WIndows除く)が使われているかを調査しなさい.

- 銀行のオンラインシステム
- 学術情報メディアセンターのスーパコンピュータ
- 携帯電話

宛先: nakashim@econ.kyoto-u.ac.jp
件名: unix2-10桁番号

- 受信通知が自動的に返信される.
- 受信通知がない場合，届いていない可能性が高い.
- メールを元にメーリングリストを作成する.

§0. 14 Cプリプロセッサ(cpp)

C言語ソースsample.cからcxxx.iを作成する.

- 注釈除去

```
/* abcd... */
```

- 識別子置換

```
#define 識別子(引数) 置換本体
#define 識別子 置換本体
#undef  識別子
```

- ファイル取り込み

```
#include <file> ... システムファイル取り込み
#include "file" ... ユーザファイル取り込み
```

- 条件付きコンパイル

```
#if     定数式
#ifdef  識別子
#ifndef 識別子
#else
#endif
```

§0. 15 Cコンパイラ本体(cc1)

cxxx.iからアセンブリ言語ソースcxxx.sを作成する.
C言語を解釈しCPU種別ごとの命令列に変換.

- 区切り
  ;
- ブロック
  {}
- 識別子
  英字or_に始まり, 英数字or_が続くもの
- 定数
  整数定数 … 10(10進数) 010(8進数)
  整数定数 … 0x10(16進数)
  浮動小数点定数 … 0.0 1.0+e10
  文字定数 … 'A' '\n'(改行) '\t'(タブ)
- 文字列
  文字並びの先頭番地… "ABC" "ABC\n"
  NULL文字(0x00)により終端されるため長さが決まる.

§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(記憶クラスに関するもの)
  static…静的 auto…自動(省略可)
  extern…外部 register…レジスタ(単なるヒント)
- 予約語(型に関するもの)
  char/int …符号付き整数(1バイト/多くは4バイト)
  short/long …符号付き整数(多くは2/4バイト)
  unsigned …符号無し整数(組合せ)
  float/double …単/倍精度浮動小数点数(通常4/8バイト)
  enum …数え上げ型
  struct …構造体
  union …共用体
  void …返り値無し関数
  typedef …型の定義
  【参考】long long …符号付き整数(8バイト)
  【参考】unsigned long long…符号無し整数(8バイト)
  【参考】long double …拡張倍精度(12or16バイト)

§0. 15 Cコンパイラ本体(cc1続き)

情報の最小単位:ビット

1ビット	□ 0または1の2通り
2ビット	□□ 00 01 10 11の4通り
1バイト(8ビット)	□□□□□□□□ 2の8乗=256通り
1ワード(16ビット)	2の16乗=6万5536通り
1ワード(32ビット)	2の32乗=42億9496万7296通り
1ワード(64ビット)	2の64乗=1844京6744兆...通り

整数の表現

8ビット符号無し	8ビット符号付き整数
00000000 ... 0	10000000 ... -128
00000001 ... 1	11111111 ... -1
00000010 ... 2	00000000 ... 0
00000011 ... 3	00000001 ... +1
11111111 ... 255	01111111 ... +127

§0. 15 Cコンパイラ本体(cc1続き)

浮動小数点数の表現(実際には32ビット以上)

8ビット浮動小数点数

符号 指数 仮数

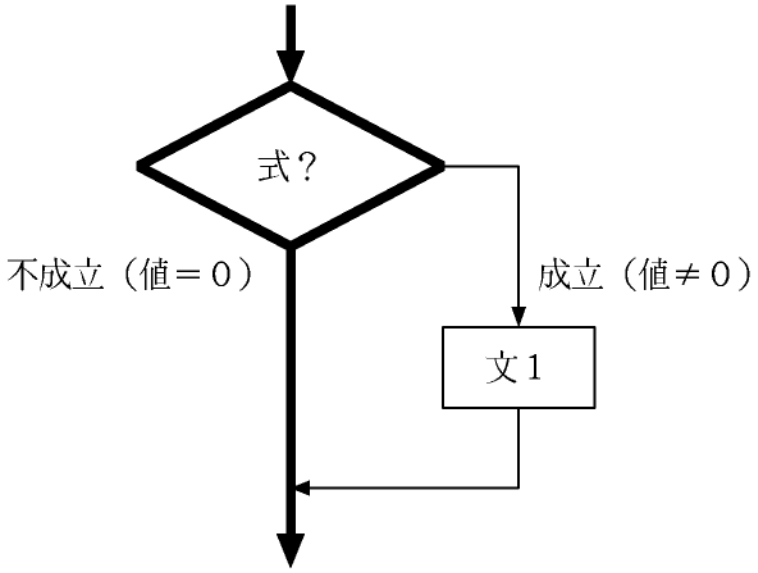
□ □□□ □□□□

0/1 111XXXX	... 非数値
0/1 1110000	... +/-∞
0/1 1100000	... +/-8
0/1 1010000	... +/-4
0/1 1000000	... +/-2
0/1 0110000	... +/-1
0/1 0100000	... +/-0.5
0/1 0010000	... +/-0.25
0/1 0000000	... +/-0

§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(条件分岐に関するもの)

```
if (式) {文1} /*式が真(!=0)のとき文1を実行*/
```

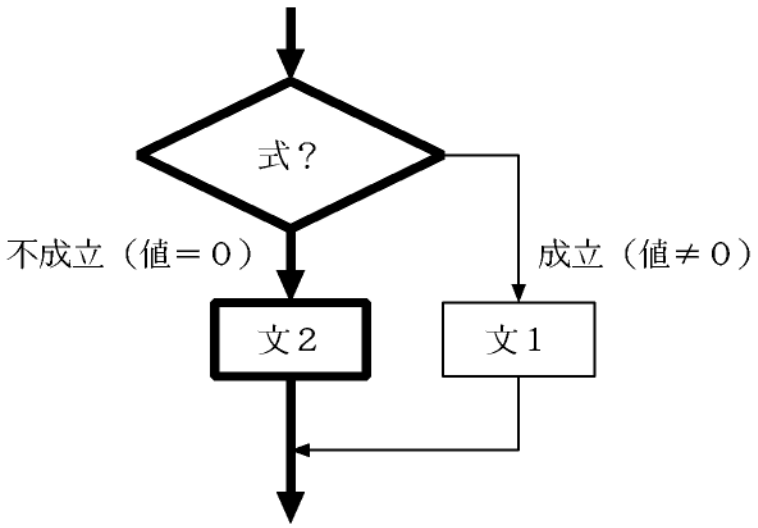



```
/* 変数aが0の場合，bに1を代入する */
if (a == 0) { b = 1; }
```

§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(条件分岐に関するもの)

```
if (式) {文1} else {文2} /*偽(==0)のとき文2を実行*/
```

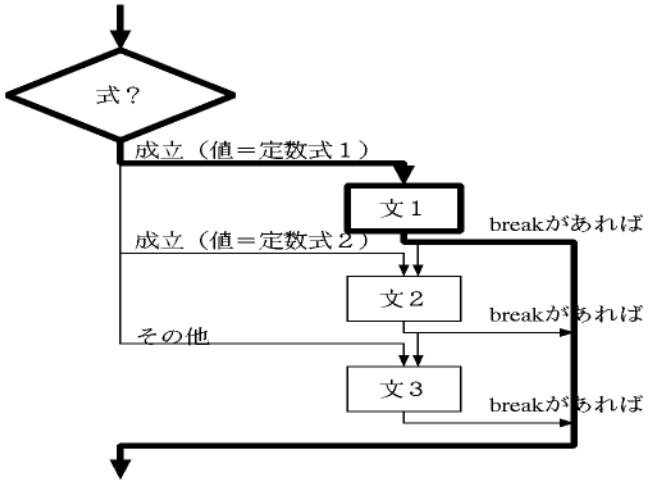



```
/* 変数aが0の場合，bに1を代入する */
/* そうでなければ，bに2を代入する */
if (a == 0) { b = 1; }
else        { b = 2; }
```

§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(条件分岐に関するもの)

```
switch (式) { /*値に応じて分岐*/
case 定数式1:
  文1
  break;       /*switchから脱出*/
case 定数式2:
  文2
  break;       /*switchから脱出*/
default:
  文3
  break;       /*switchから脱出*/
}
```

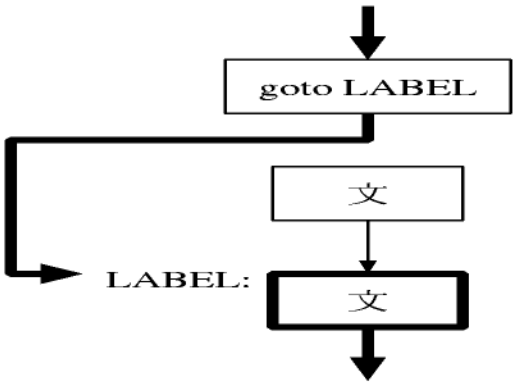



```
switch (a) {
case 0:
  b = 1; break;
case 1:
  b = 2; break;
default:
  b = 3; break;
}
```

§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(無条件分岐に関するもの)

```
goto 識別子; /*識別子へ分岐する*/
識別子:
```

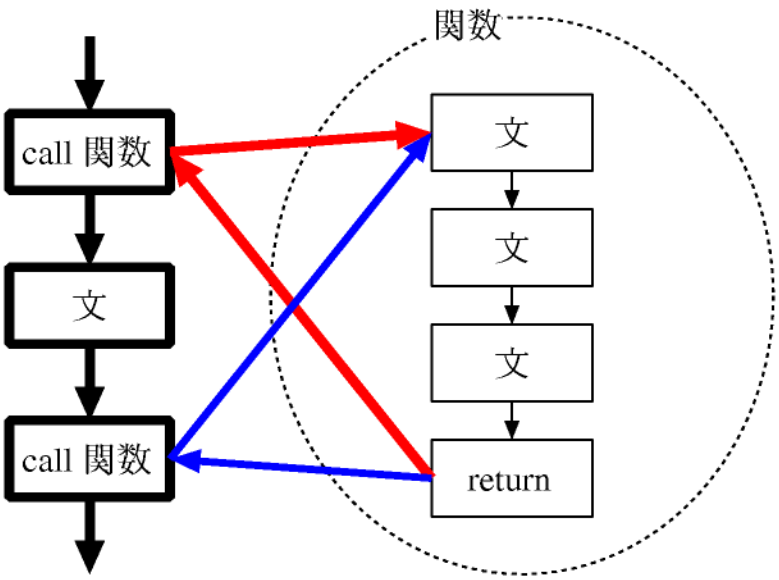



```
if (a == 0) {
 b = 1;
 goto handan0;
}
b = 2;
handan0:
```

§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(無条件分岐に関するもの)

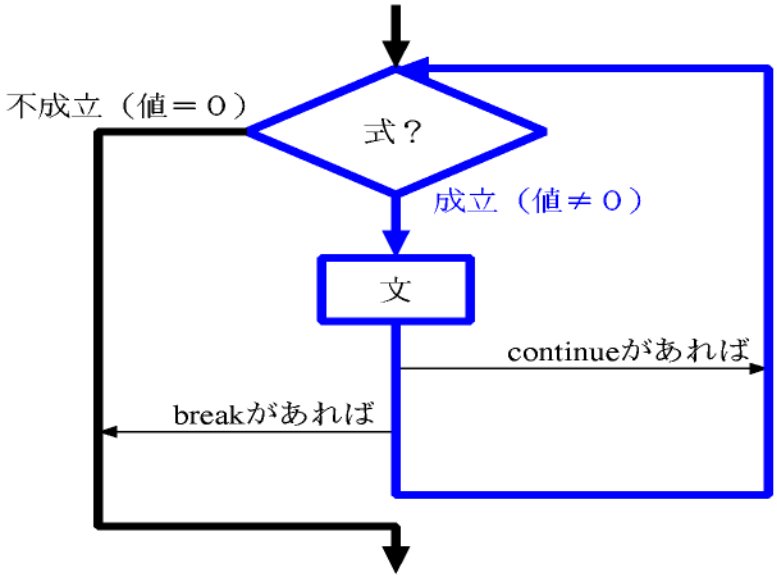
```
return; /*関数を終了し呼出し元へ戻る*/
```



§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(ループ構造に関するもの)

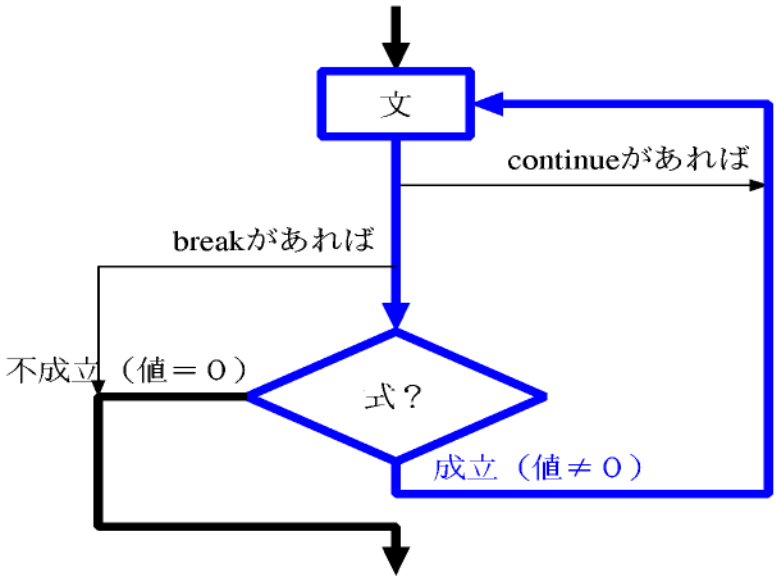
```
while (式) { /*式が真(!=0)の間繰り返す*/
  文
  continue;   /*whileの最後に分岐*/
  break;      /*whileから脱出*/
}
```



§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(ループ構造に関するもの)

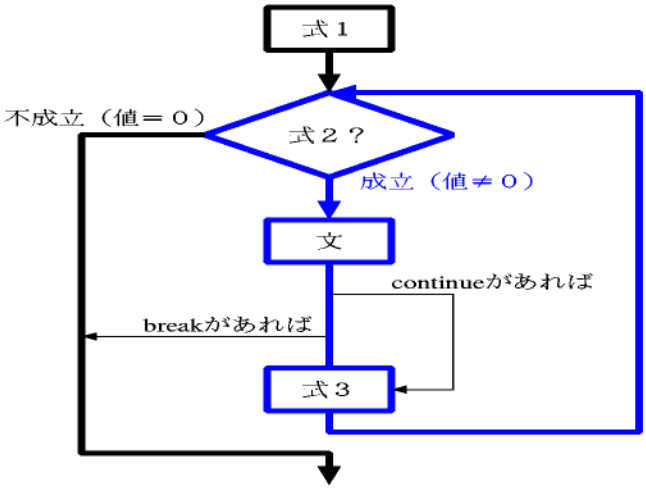
```
do {
  文
  continue;   /*doの最後に分岐*/
  break;      /*doから脱出*/
} while (式) /*式が真(!=0)の間繰り返す*/
```



§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(ループ構造に関するもの)

```
for (式1;式2;式3) { /*最初に1回だけ式1を実行*/
          /*式2が真(!=0)の間, 最後に式3を実行し繰り返す*/
  文
  continue; /*forの最後に分岐*/
  break;    /*forから脱出*/
}
```



§0. 15 Cコンパイラ本体(cc1続き)

- 予約語(式の一部として用いるもの)

```
sizeof
```

- 予約語(その他)

本講義では取り挙げないが，識別子として用いることはできない

```
asm inline typeof const volatile signed
```

§0. 15 Cコンパイラ本体(cc1続き)

- 演算子(優先順位の高い順)

結合規則		
左から右	`() [] -> .`	関数,配列,ポインタ,構造体
右から左	`! ~ ++ -- - (type) * & sizeof`	単項演算子
左から右	`* / %`	乗法演算子
左から右	`+ -`	加法演算子
左から右	`<< >>`	シフト演算子
左から右	`< <= > >=`	関係演算子
左から右	`== !=`	等値演算子
左から右	`&`	ビット論理積演算子
左から右	`^`	ビット排他論理和演算子
左から右	`|`	ビット論理和演算子
左から右	`&&`	論理積演算子
左から右	`||`	論理和演算子
右から左	`?:`	条件演算子
右から左	`= *= /= %= += -= <<= >>= &= ^= |=`	代入演算子
左から右	`,`	コンマ演算子

今日はここまで