# プログラミング基礎

2022年度1Q　火曜日３,４時限(10:45~12:25)
金曜日1~４時限(　8:50~12:25)

工学院　情報通信系

中山実，渡辺義浩
伊藤泉，小杉哲
TA:小泊大輝，千脇彰悟

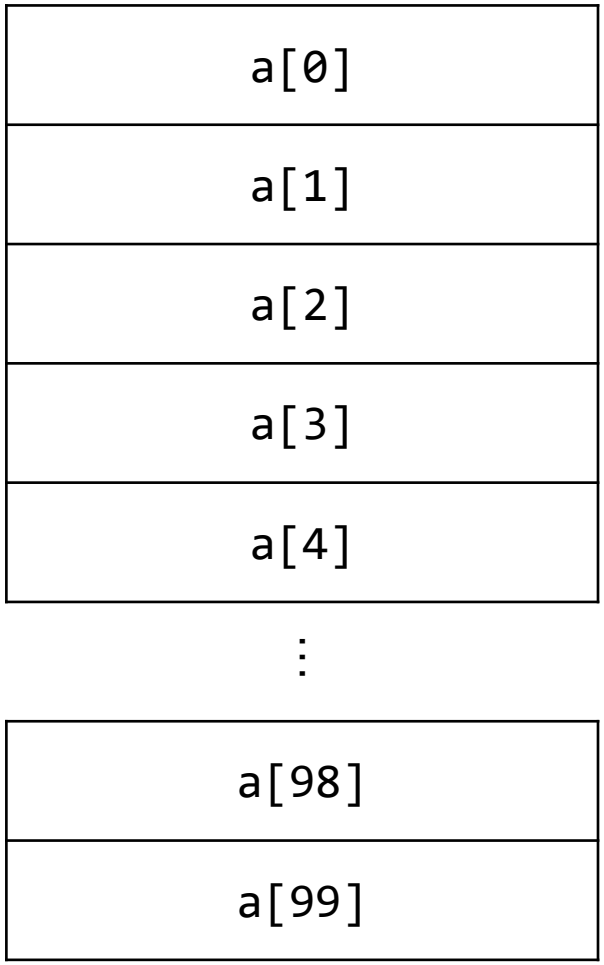
# 4/25(火) 10:45~12:25

- 第5回「配列１」

1. 配列の宣言と初期化
2. 配列の使い方
3. 文字配列

# 配列（array）

- `int a[100];`
  - メモリ上の連続領域にint型変数100個分の領域が確保される
- `a[0], a[1], …, a[99]`
  - 添え字（index）を使って各領域を指定できる
  - 注意：０から始まる！
- `a[10*i+j]`
  - 添え字は数式でもいい.



# 配列の宣言と初期化

```c
#include <stdio.h>

int main(void){
    int i;
    int a[6]={2, 5, 3, 5, 6, 1};

    for(i=0; i<6; i++){
        printf("%d ", a[i]);
    }
    return 0;
}
```

実行結果

```
2 5 3 5 6 1
```

# 配列の宣言と初期化

```
int a[6]={2, 5, 3, 5, 6, 1};
```

- 単独の変数の時と同様に，宣言と同時に初期値を与えることができる.
- 要素の数だけ初期値を並べる.

- 宣言の数より要素が足りないと0が入力される.
- a[6]={2，5，3};として実行してみよう.

# 配列の宣言と初期化

```
int a[6]={2, 5, 3, 5, 6, 1};
```

- 宣言の数を入力しないと初期値の数に合わせて宣言される.
- a[]={2, 5, 3, 5, 6, 1};でも可能.

- 宣言の数より多く初期値を入れてもコンパイラは認識できない.
- 暴走の可能性があるので十分に注意する. (試してみてもいい)

# 配列の利用

```c
#include <stdio.h>

int main(void) {

    int i;
    int a[11];

    for (i=0;  i<=10; i++){
        a[i] = i*i;
    }
    for (i=10 ; i>=0;  i--){
        printf ("%d ", a[i]);
    }
    return 0;
}
```

- 11個の要素を持つ配列を宣言

実行結果

```
100 81 64 49 36 25 16 9 4 1 0
```

# 配列の利用

```c
#include <stdio.h>

int main(void) {

    int i;
    int a[11];

    for (i=0;  i<=10; i++){
        a[i] = i*i;
    }
    for (i=10 ; i>=0;  i--){
        printf ("%d ", a[i]);
    }
    return 0;
}
```

- 0~10までの数値の二乗を配列に格納する.

実行結果

```
100 81 64 49 36 25 16 9 4 1 0
```

# 配列の利用

```c
#include <stdio.h>

int main(void) {

    int i;
    int a[11];

    for (i=0;  i<=10; i++){
        a[i] = i*i;
    }
    for (i=10 ; i>=0;  i--){
        printf ("%d ", a[i]);
    }
    return 0;
}
```

実行結果

```
100 81 64 49 36 25 16 9 4 1 0
```

- 配列の要素を大きい方から順に出力
- この例では，配列を使う必然性はなく，出力時に計算してもよい
- 実際には，一旦配列にデータを格納し，あとで出力させる場合も多い

# 文字型配列の宣言と初期化

- 文字型配列の宣言
  - 一番最後に文字列の終わりを示す“¥0”(＼0)を入れる
  - エスケープシーケンス，NULL文字
  - “¥0”も考えた長さにする.

```
#include <stdio.h>

int main(void) {
    char str[6]={'H','e','l','l','o','¥0'};

    printf("%c¥n", str[0]);
    printf("%s¥n ", str);

    return 0;
}
```

実行結果

```
H
Hello
```

# 文字型配列の宣言と初期化

- ほかの宣言と初期化例
  - 文字列の場合は，" "とする.

```
char str[]={'H','e','l','l','o','¥0'};
char str[6]="Hello";
char str[]="Hello";
```

# 文字型配列の宣言と初期化

- 後で代入は不可.

```c
#include <stdio.h>

int main(void) {
    char str[6]={'H','e','l','l','o','¥0'};

    str ="Hello";

    printf("%c¥n", str[0]);
    printf("%s¥n ", str);

    return 0;
}
```

# 文字型配列の宣言と初期化

- 1文字の出力：%c
- 文字列全体の出力：%s

```
#include <stdio.h>

int main(void) {
    char str[6]={'H','e','l','l','o','¥0'};

    printf("%c¥n", str[0]);
    printf("%s¥n ", str);

    return 0;
}
```

実行結果

```
H
Hello
```

# 文字型配列の宣言と初期化

- 大きい配列を宣言
- 文字の入力．変換指定子は%s
- 変数(str)の前に&をつけない．

```c
#include <stdio.h>

int main(void) {

    char str[100];

    printf("Input a word¥n");
    scanf("%s", str);
    printf("The word is: %s ¥n", str);

    return 0;
}
```

実行結果

```
Input a word
Happy    //入力した文字
The word is: Happy
```

# 配列への読み込み

```c
#include <stdio.h>

int main(void){
    int a[3];

    scanf("%d", a);
    scanf("%d", a+1);
    scanf("%d", &(a[2]));

    for(int i=0; i<3; i++)
        printf("%d ", a[i]);
    printf("¥n");

}
```

実行結果

```
1
2
3
1 2 3
```

# printf 書式 (再掲)

- %[flags][width][ . ][precision][typeのプレフィックス]type

| フィールド | 指定値 | 意味 |
|---|---|---|
| flags | - | 変換される引数を左詰めにします。指定しないと右詰めにします。 |
| | + | 出力する値が符号付きの場合「+」または「-」の符号をつけます。指定しないとマイナスのときのみ「-」がつきます。 |
| | 0 | 最小幅になるまで0を追加します。出力値がマイナスと0の場合は0は追加されません。省略すると0は追加されません。 |
| width | 数値 | 文字幅を指定します。表示する文字数が大きい場合は自動で拡張します。 |
| precision | 数値 | 精度を指定します。つまり小数点以下の桁数を指定します。 |
| Typeのプレフィックス | h | Short intに適応させます。 |
| | l | long修飾です。 |
| | L | long double修飾です。 |

[粂井2013]

# printf 書式 (再掲)

- %[flags][width][ . ][precision][typeのプレフィックス]type

| フィールド | 指定値 | 意味 |
|---|---|---|
| type | d,i | 10進整数を指定します。 |
| | o | 符号なし8進数を指定します。 |
| | u | 符号なし10進整数を指定します。 |
| | x | 符号なし16進整数で、a，b，c，d，e，fを使います。 |
| | X | 符号なし16進整数で、A，B，C，D，E，Fを使います。 |
| | f | 浮動小数点を指定します。 |
| | e，E | 指数変換を指定します。±〇.〇〇e〇〇または±〇.〇〇E〇〇。 |
| | c | 文字変換を指定します。 |
| | s | 文字列変換を指定します。 |
| | p | ポインタ変換を指定します。 |
| | g | 指数の絶対値が大きい時に指数表示で出力 |

[粂井2013]

# 演算子一覧 (再掲)

| 優先順位 | 記号 | 意味 |
|---|---|---|
| 1 | () | 関数呼び出し。printf(…), main()等 |
| | [] | 配列 |
| | -> | 構造体メンバ参照 |
| | . | 構造体メンバ参照 |
| | ++ | 後置増分 |
| | -- | 後置減分 |
| 2 | ++ | 前置増分 |
| | sizeof | 記憶量 |
| | & | アドレス |
| | * | 間接参照 |
| | + | 正符号 |
| | - | 負符号 |
| | ~ | 1の補数(ビット反転) |
| | ! | 否定 |

| 優先順位 | 記号 | 意味 |
|---|---|---|
| 3 | () | キャスト |
| 4 | * | 乗算 |
| | / | 除算 |
| | % | 剰余 |
| 5 | + | 加算 |
| | - | 減算 |
| 6 | << | ビット左シフト |
| | >> | ビット右シフト |
| 7 | < | 左不等号（大きい） |
| | <= | 等価左不等号（以上） |
| | > | 右不等号（小さい） |
| | >= | 等価右不等号（以下） |
| 8 | == | 等価 |
| | != | 非等価 |

# 演算子一覧 (再掲)

| 優先順位 | 記号 | 意味 |
|---|---|---|
| 9 | & | ビットごとの論理積 |
| 10 | ^ | ビットごとの排他的論理和 |
| 11 | \| | ビットごとの論理和 |
| 12 | && | 論理積 |
| 13 | \|\| | 論理和 |
| 14 | ?: | 条件。a?b:cのように使う |
| 15 | = | 代入 |
| | += | 加算代入 |
| | -= | 減算代入 |
| | *= | 乗算代入 |
| | /= | 除算代入 |
| | %= | 剰余代入 |
| | <<= | 左シフト代入 |
| | >>= | 右シフト代入 |

| 優先順位 | 記号 | 意味 |
|---|---|---|
| 15 | &= | ビット積代入 |
| | ^= | ビット差代入 |
| | \|= | ビット和代入 |
| 16 | , | コンマ演算子 |

# “math.h”関数の一部

| 戻り値 | 関数 | 処理の内容 |
|---|---|---|
| double | cos(double x) | コサインを返す |
| double | sin(double x) | サインを返す |
| double | tan(double x) | タンジェントを返す |
| double | acos(double x) | アークコサインを返す |
| double | asin(double x) | アークサインを返す |
| double | atan(double x) | アークタンジェントを返す |
| double | atan2(double y, double x) | y/xのアークタンジェントを返す |
| double | cosh(double x) | 双曲線コサインを返す |
| double | sinh(double x) | 双曲線サインを返す |
| double | tanh(double x) | 双曲線タンジェントを返す |

# “math.h”関数の一部

| 戻り値 | 関数 | 処理の内容 |
|---|---|---|
| double | ceil(double x) | xより大きい最も小さい整数を返す |
| double | floor(double x) | xより小さい最も大きい整数を返す |
| double | fabs(double x) | 絶対値を返す |
| double | pow(double x, double y) | べき乗(xのy乗)を返す |
| double | sqrt(double x) | 平方根を返す |
| double | log(double x) | 自然対数を返す |
| double | exp(double x) | 指数関数$e^x$を返す |

# 参考 (再掲)

- 繰り返し処理が終了しない時(無限ループに入ってしまった場合)
  - 実行窓の■赤ボタン（左上）を押す.

- Mac OSの場合：
  - バックスラッシュ(\)：option + ¥
  - 無限ループに入る，画面が固まる：option + command + esc（強制終了）

# 参考

- 無限ループの例

```
while (1) {
   …;
}

do {
   …;
} while (1) ;
```