プログラミング入門
第11回

KITAMI
Institute of Technology

関数のしくみを知る(pp.216-217)

- 関数(function)
  - 一定の処理をまとめて記述したものを「関数」でまとめる

$x^2 + x + 1$

$f(x) = x^2 + x + 1$ 関数を定義

$x = 1$のとき
$1^2 + 1 + 1 = 3$

$f(1) = 3$

関数を呼び出す

$x = 2$のとき
$2^2 + 2 + 1 = 7$

$f(2) = 7$

関数のしくみを知る(pp.216-217)

- 関数(function)
  - C言語で関数を記述した場合
  - 何度でも呼び出すことができる

```
int main (void)
{
   int x, y;
   x = 1;
   y = x*x+x+1;
   x = 2;
   y = x*x+x+1;
...
```

```
int f(int x){
 return x*x+x+1;
}
```

関数を定義

```
int main(void)
{
     int y;
     y = f(1);
     y = f(2);
...
```

関数を呼び出す

関数を定義する(p.218-219)

- 関数の定義

```
戻り値の型　関数名（引数リスト）
{
        文;
        ....
        return 式;
}
```

使用しない場合もある

```
void buy(void) {
        printf("車を買いました¥n");
}
```

関数の入力の値や結果の出力が無いときに
voidを使う（後述）

関数を呼び出す(p.219-220)

- 関数の呼び出し

```
関数名(引数リスト);
```

引数を使用しない場合は()とする

```
buy();
```

最後にセミコロンを付ける

関数を呼び出す(p.220-221)

- Sample1.cのソースコード

```
#include <stdio.h>

/* buy関数の定義 */
void buy(void)
{
  printf("車を買いました。¥n");
}

/* buy関数の呼び出し */
int main(void)
{
  buy();

  return 0;
}
```

③ buy()関数の内部処理を行う

① ② buy()関数を呼び出す

④ 呼び出し元に戻る

関数を何度も呼び出す(p.222-223)

- 関数を2度呼び出した場合

```
void buy(void)
{
  printf("車を買いました。¥n");
}
```

```
int main(void)
{
  buy();  //「車を買いました」と表示する関数buy()の呼び出し
  printf("もう1台車を購入します。¥n");
  buy();  //「車を買いました」と表示する関数buy()の呼び出し
  return 0;
}
```

実行結果

```
車を買いました。
もう1台車を購入します。
車を買いました。
```

関数に処理をまとめる(p.224-225)

- 関数で複雑の文を処理させる場合

```
void buy(void)
{
   printf("1台目の車を買いました。¥n");
   printf("もう1台車を購入する。¥n");
   printf("2台目の車を買いました。¥n");
}
```

```
int main(void)
{
   buy();  // 3行分の表示処理を行う
}
```

引数(p.226-227)

- 引数(argument)
  - 関数を呼び出すときに情報(値)を引き渡すことができる

int型の引数を用意する

```
/* buy関数の定義 */
void buy( int x )
{
  printf("%d万円の車を買いました。¥n", x);
}
```

引数を関数内で使う

引数(p.227-228)

- 関数内で値を受け取る変数を仮引数という
- 関数で呼ぶ出す際に渡す値を実引数という

```
void buy( int x )
{
   printf("%d万円の車を買いました。¥n", x);
}

int main(void)
{
   buy(20);
   buy(50);
   return 0;
}
```

仮引数

実引数

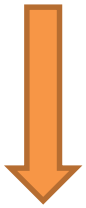
引数(p.228-229)

- 処理の流れ
  - buy()関数を呼び出すときに実引数「20」を渡す
  - 関数内で仮引数「20」で表示処理する
  - buy()関数を呼び出すときに実引数「50」を渡す
  - 関数内で仮引数「50」で表示処理する

複数の引数をもつ関数を使う (p.232-233)

- 引数リスト
  - 複数の実引数をカンマ(,)で区切る
  - 区切った順に実引数の値が仮引数に渡される

```
buy(num1, num2);
```



実引数num1の値を仮引数xに渡す

実引数num2の値を仮引数yに渡す

```
void buy( int x, int y )
{
  printf("%d万円と%d万円の車を買いました。¥n", x, y);
}
```

引数のない関数を使う(p.234-235)

- 引数のない関数を定義するときには、引数の型としてvoidを使う
  - void (空の, 空虚な, 無効の)

```
void buy(void)
{
  printf("車を買いました。¥n");
}
```

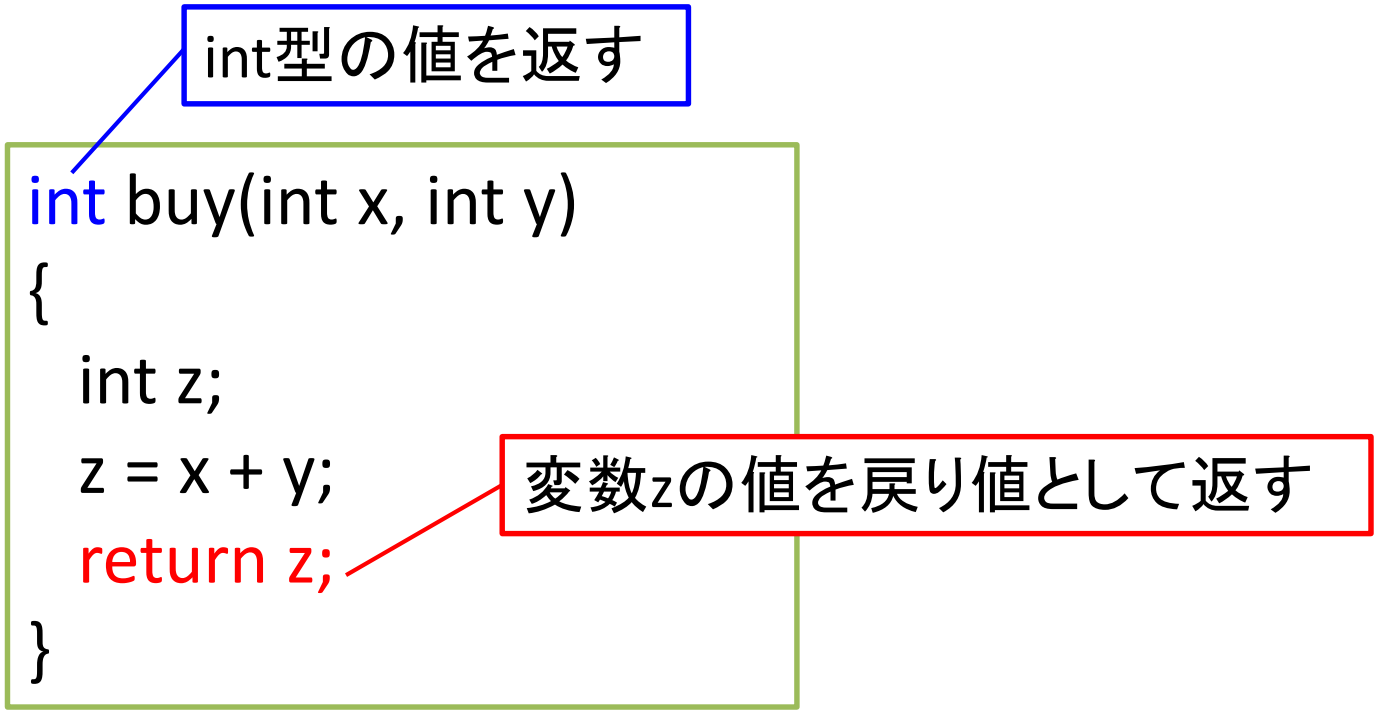
引数を持たない場合はvoid型とする

```
int main (void)
{
  buy();
  return 0;
}
```

引数を渡さないで関数を呼び出す

戻り値のしくみを知る(p.236-237)

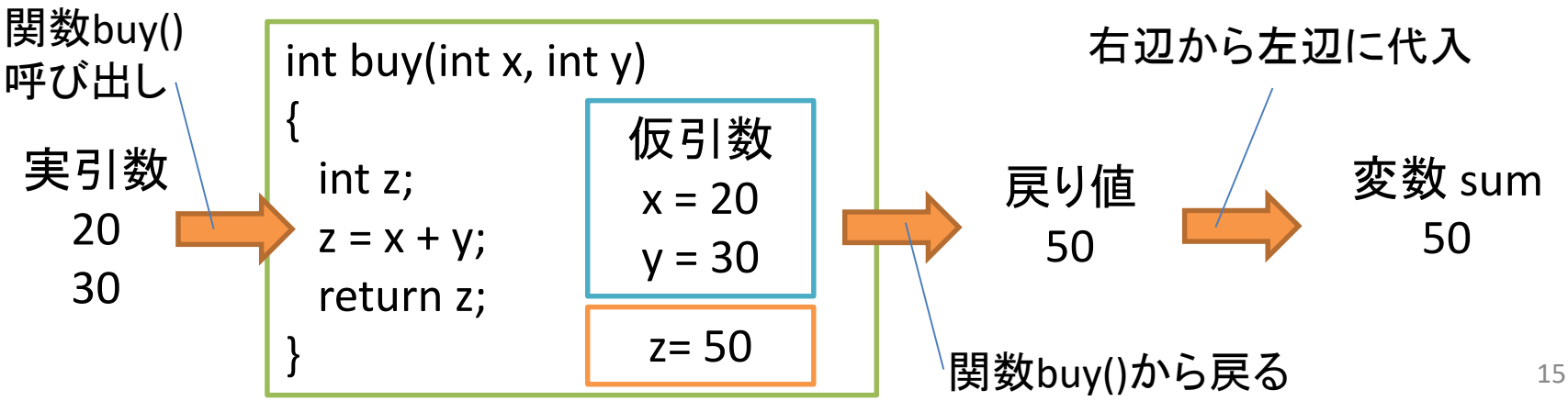
- 関数の呼び出し元に関数本体から特定の情報を返す
  - returnという文を使って値を返す処理を行う



戻り値のしくみを知る(p.237-238)

- 関数内で計算された結果を戻り値として変数に格納する

```
int main(void)
{
  int sum;
  sum = buy(20, 30);
  return 0;
}
```



戻り値のない関数(p.239-240)

- 関数が戻り値をもたない場合は戻り値の型として void型を指定する
- 何もつけないreturn文によって関数が終了する
  - return文は省略しても同じ

```
void buy(void)
{
  printf("車を買いました。¥n);
  return;
}
```

もしくは

```
void buy(void)
{
  printf("車を買いました。¥n);

}
```

関数の利用(p.242-243)

- 最大値を求める関数
  - 変数xの値がyより大きいときはxの値を返す
  - それ以外はyの値を返す
    - xとyの値が等しいときも含む

```
int max(int x, int y)
{
   if (x > y)
      return x;
   else
      return y;
}
```

必ず大きいのほうの値を返す

main()関数の意味(p.217)

- C言語で書かれたプログラムはmain()関数から実行される
- C言語規格としてint型の戻り値を返すのが慣例である

```
int main(void)
{
   …
return 0;
}
```

```
int main() // voidを省略する場合もある
{
   …
return 0;
}
```

プログラムが正常終了したときの合図として0を返す

（注）引数リストはvoidの代わりに仮引数を与える場合もある
(p. 421参照)