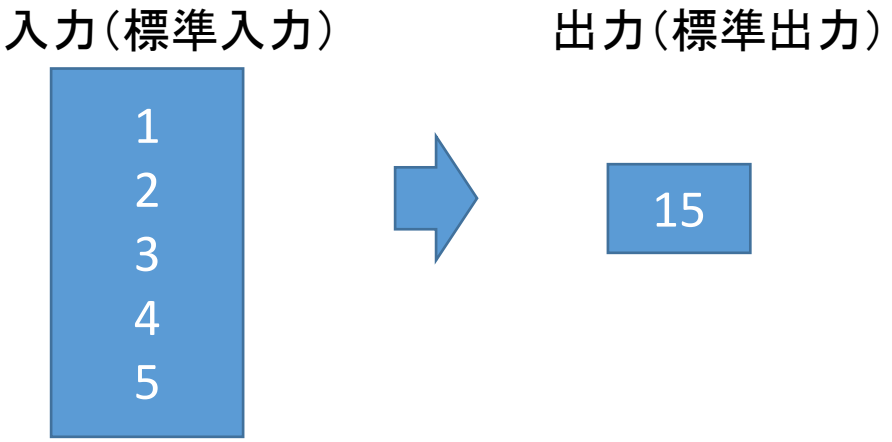
初歩の初歩

- 5個の整数が与えれられたとき、その和を計算して出力するプログラムを作れ

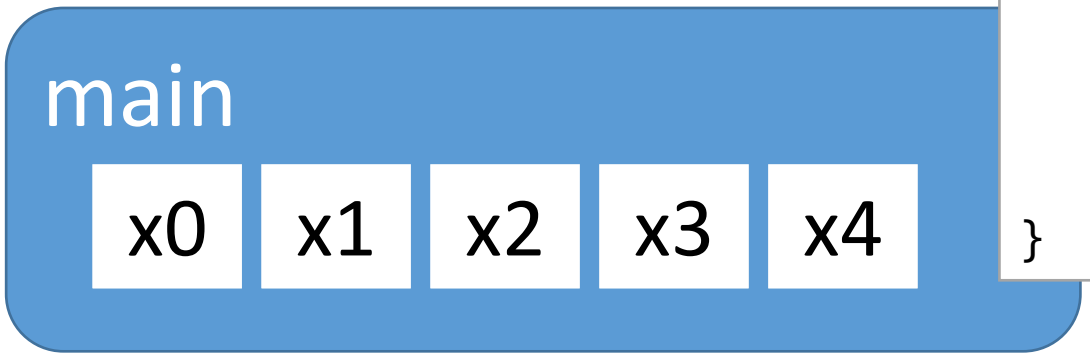


初歩の初歩

- 「繰り返し」なしで解いてみる
- 変数 x0, x1, …, x4

C++ の場合

```
int main(void)
{
    int x0, x1, x2, x3, x4;
    cin >> x0;
    cin >> x1;
    cin >> x2 >> x3 >> x4;
    cout << (x0 + x1 + x2 +
             x3 + x4) << endl;
    return 0;
}
```



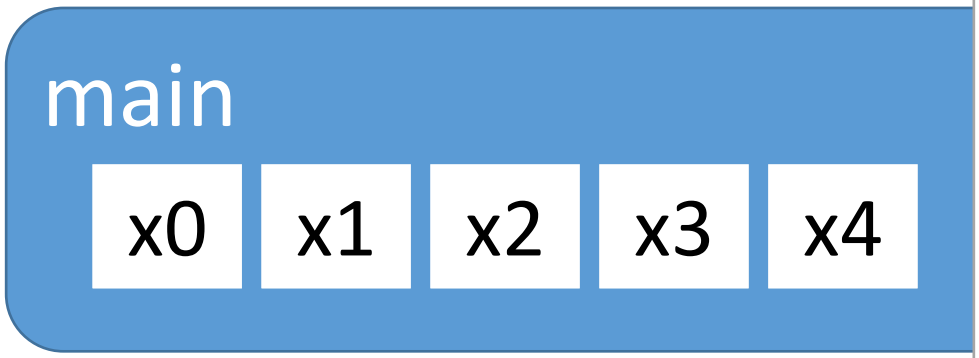
プログラムは、以下に配置：
https://github.com/tomiokamada/regio/tree/master/samples

初歩の初歩

- 「繰り返し」なしで解いてみる
- 変数 x0, x1, …, x4

C の場合

```c
int main(void)
{
    int x0, x1, x2, x3, x4;
    scanf("%d", &x0);
    scanf("%d", &x1);
    scanf("%d", &x2);
    scanf("%d", &x3);
    scanf("%d", &x4);
    printf(“%d¥n”,
            x0+x1+x2+x3+x4);
    return 0;
}
```



デモンストレーション

- デバッガで、プログラム動作の様子を眺めてみましょう。
  - プログラムの動作イメージが視覚的に描けることは重要
  - 思った通りにプログラムが動作しているか、確かめることもできる

初歩の初歩

- 「繰り返し」なしで解いてみる
- 変数
  - x 入力用
  - s 計算結果用


```
int main(void)
{
    int x, s=0;
    cin >> x;
    s += x;
    cin >> x;
    s += x;
    cin >> x;
    s += x;
    cin >> x;
    s += x;
    cin >> x;
    s += x;
    cin >> x;
    s += x;
    cout << s << endl;
    return 0;
}
```

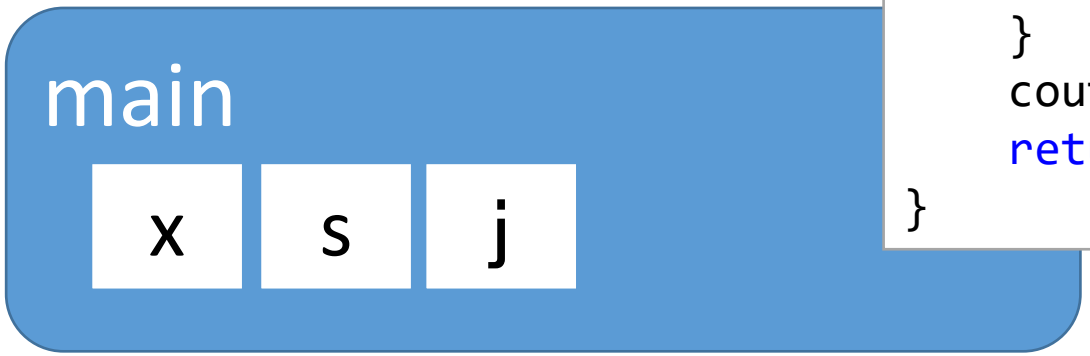
初歩の初歩

- 「繰り返し」なしで解いてみる
- 変数
  - x 入力用
  - s 計算結果用


```c
int main(void)
{
    int x, s=0;
    scanf("%d", &x);
    s += x;
    scanf("%d", &x);
    s += x;
    scanf("%d", &x);
    s += x;
    scanf("%d", &x);
    s += x;
    scanf("%d", &x);
    s += x;
    printf("%d¥n", s);
    return 0;
}
```

初歩の初歩

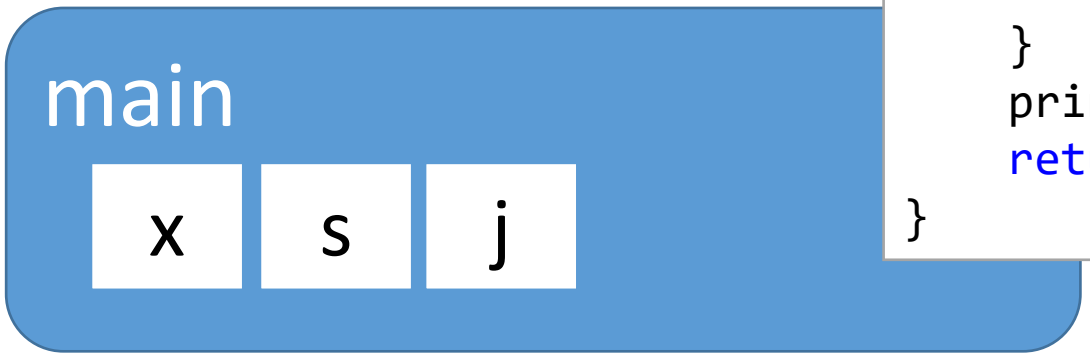
- 「繰り返し」を使ってみる
- 変数
  - x 入力用
  - s 計算結果用
  - j 回数カウント用




```
int main(void)
{
    int x, s=0;
    for (int j = 0; j<5; j++) {
        cin >> x;
        s += x;
    }
    cout << s << endl;
    return 0;
}
```

初歩の初歩

- 「繰り返し」を使ってみる
- 変数
  - x 入力用
  - s 計算結果用
  - j 回数カウント用

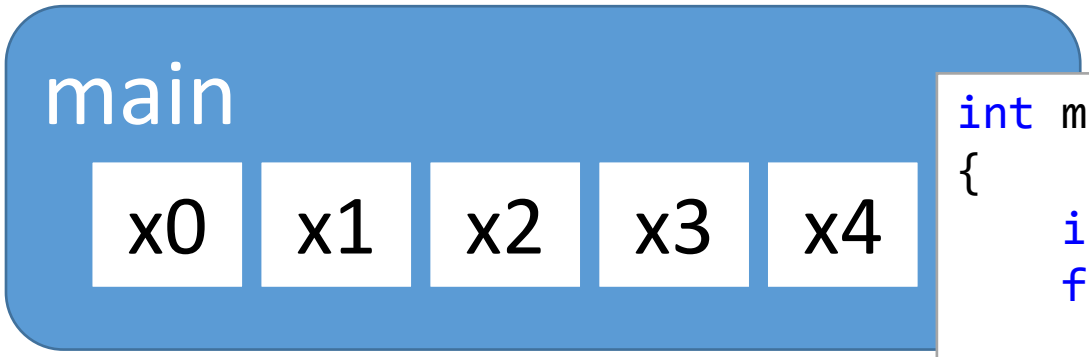



```c
int main(void)
{
    int x, s=0, j;
    for(j=0; j<5; j++) {
        scanf("%d", &x);
        s += x;
    }
    printf("%d¥n", s);
    return 0;
}
```

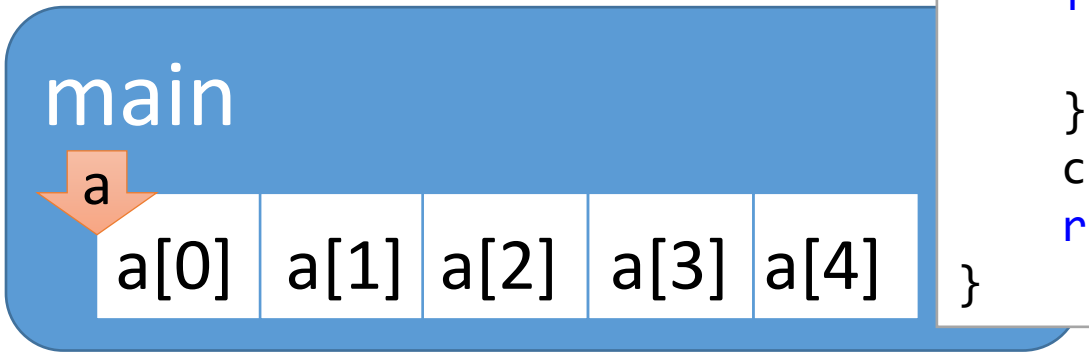
追加で配列のイメージ

- 2回ループを回す必要は「ない」
- でも、先にデータを取り込んだ方がわかりやすいことも多い
- 分かりやすいのが一番
- 大きな配列は局所変数より大域変数で

■ 以前 : 変数 x0, x1, …, x4



■ 配列: int a[5]

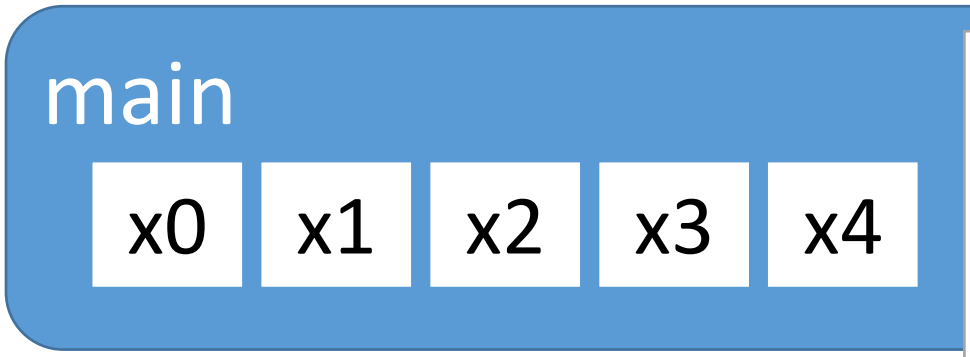



```
int main(void)
{
    int a[5], s=0;
    for (int j = 0; j < 5; j++) {
        cin >> a[j];
    }
    for (int j = 0; j < 5; j++) {
        s += a[j];
    }
    cout << s;
    return 0;
}
```

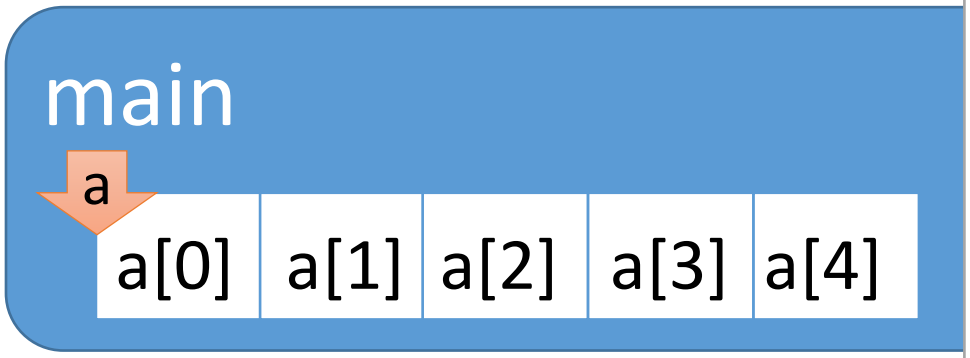
追加で配列のイメージ

- 2回ループを回す必要は「ない」
- でも、先にデータを取り込んだ方がわかりやすいことも多い
- 分かりやすいのが一番
- 大きな配列は局所変数より大域変数で

■ 以前：変数 x0, x1, …, x4



■ 配列: int a[5]

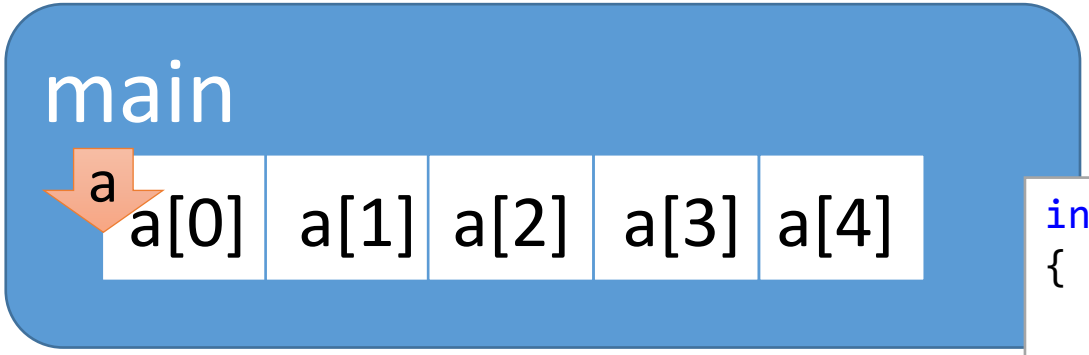



```c
int main(void)
{
    int a[5], j, sum=0;
    for (j = 0; j < 5; j++) {
        /* とりあえず a[0]-a[4]に
           値を格納してみる */
        scanf("%d", &a[j]);
    }
    for (j = 0; j < 5; j++) {
        sum += a[j];
    }
    printf("%d¥n", sum);
    return 0;
}
```

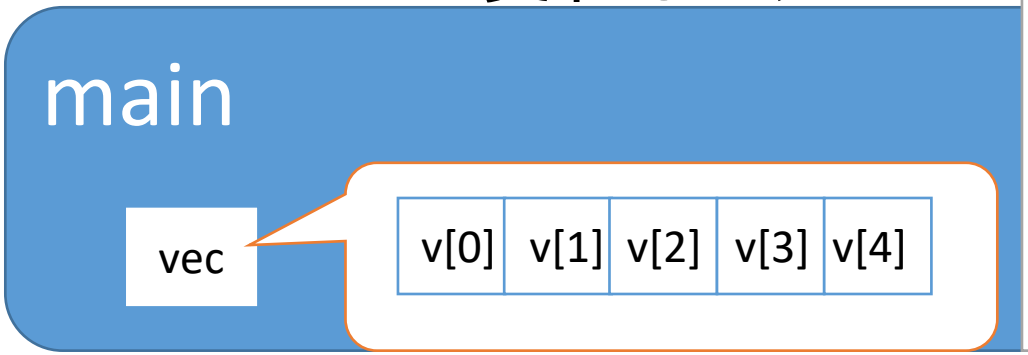
C 言語版

追加で配列のイメージ (vector)

- 配列: int a[5]



- vector: vector<int> vec;
  サイズが変わる配列的




```
int main(void)
{
    vector<int> v;
    for (int j = 0; j < 5; j++) {
        cin >> v[j];
    }
    int s = 0;
    for (int j = 0; j < 5; j++) {
        s += v[j];
    }
    cout << s << endl;
    return 0;
}
```

ミッション

プログラム置き場
https://github.com/tomiokamada/regio/tree/master/debug

- デバッガで実際にバグを見つけてみましょう
  - 1 + 1 / 2 + 1 / 3 + … + 1 / 9 を求めるプログラム。
  - 実行すると、結果は 1

debugTest.cpp

```
int main(void)
{
    int i;
    float result = 0.0;
    for (i = 1; i < 10; i++)
    {
        float tmp = 1 / i;
        result = result + tmp;
    }
    cout << "result = " << result << endl;
    return 0;
}
```