

ca-6. プログラムカウンタ

(コンピュータ・アーキテクチャ演習)

URL: <https://www.kkaneko.jp/cc/ca/index.html>

金子邦彦



アウトライン



6-1 プログラムカウンタの振る舞い

6-2 Visual Studio でプログラムカウンタの表示

6-3 命令実行サイクル

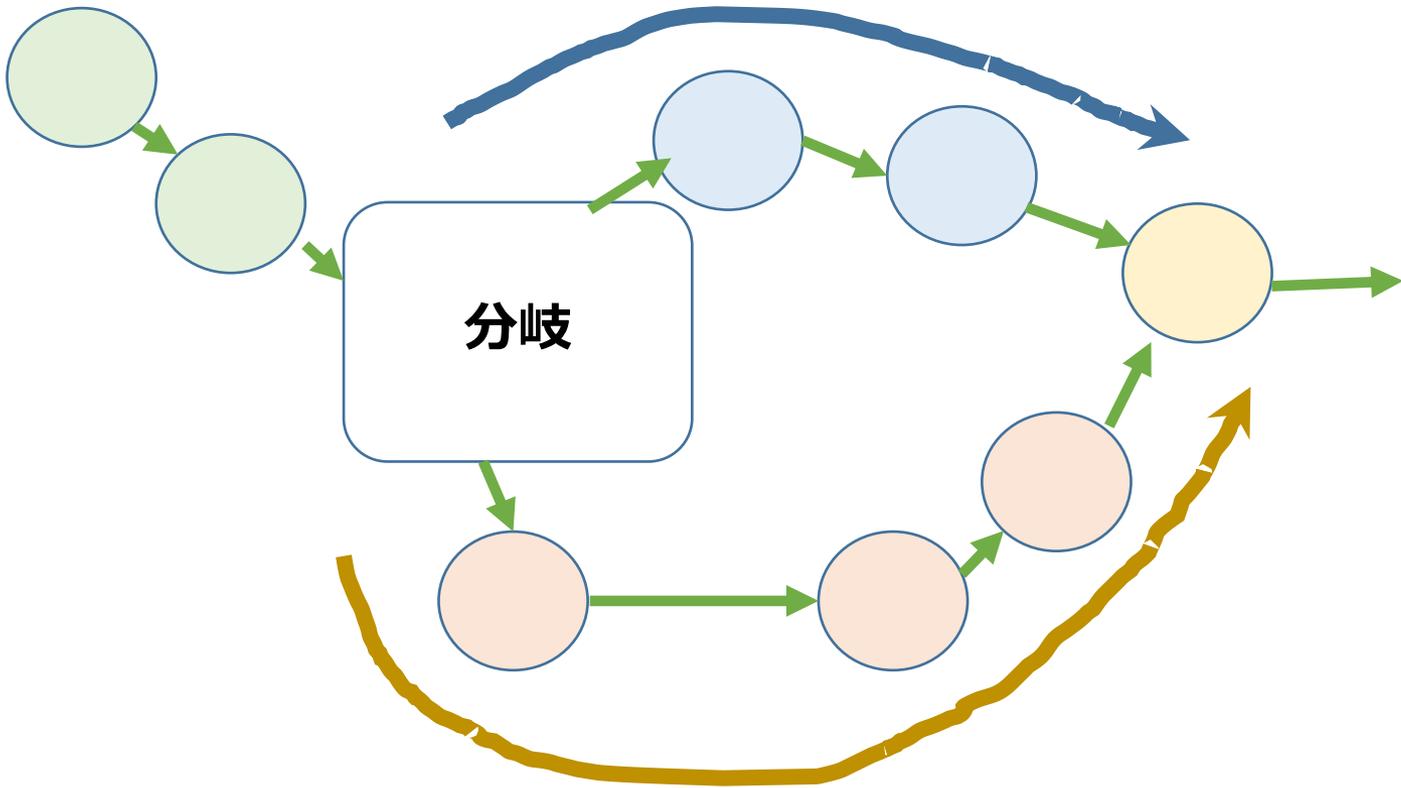
6-1 プログラムカウンタの振る舞い

6-1 プログラムカウンタとは



- プログラムカウンタには、次に実行すべき命令のアドレスが入っている

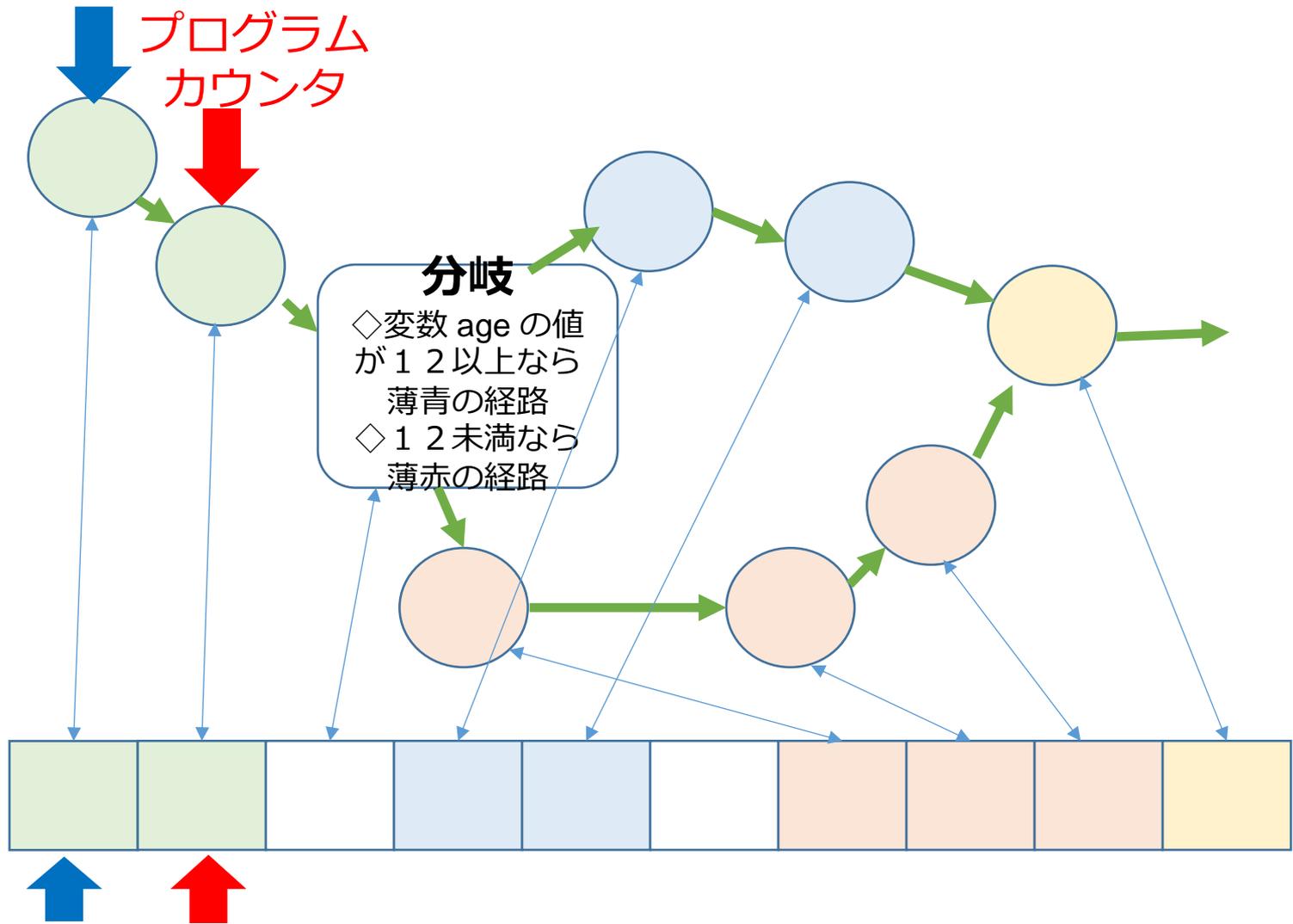
変数 age の値が 12 以上の
ときの経路

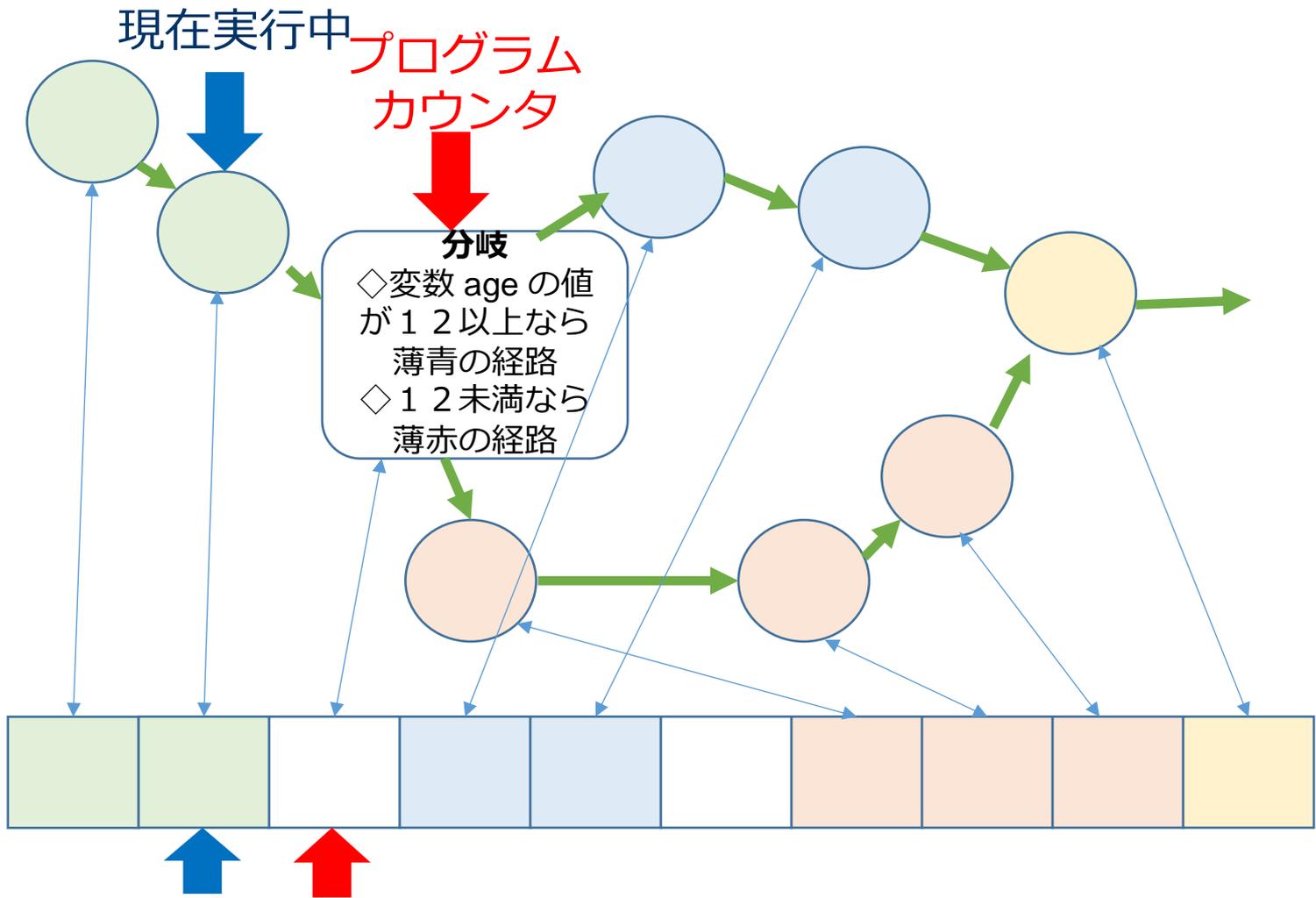


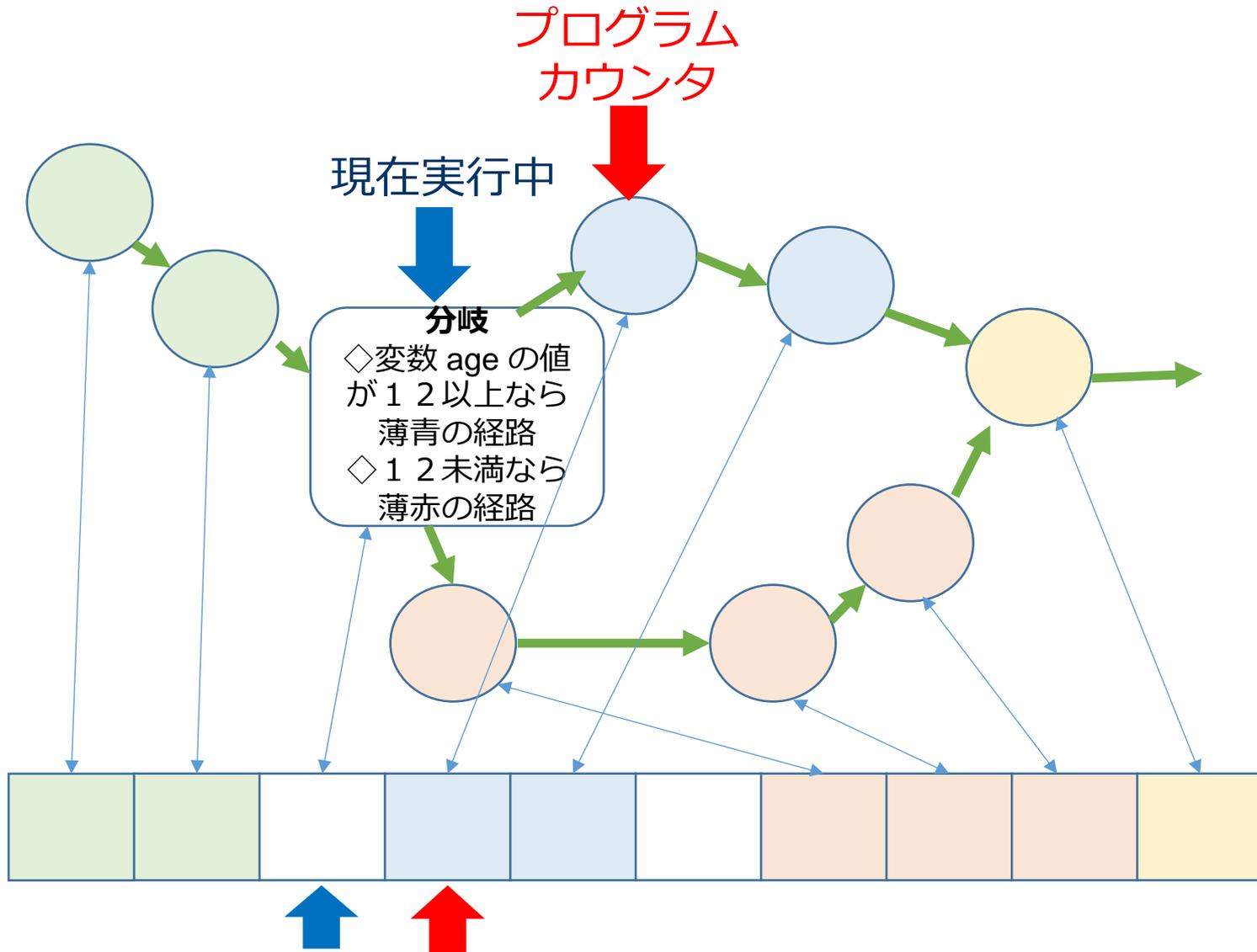
変数 age の値が 12 未満の
ときの経路

現在実行中

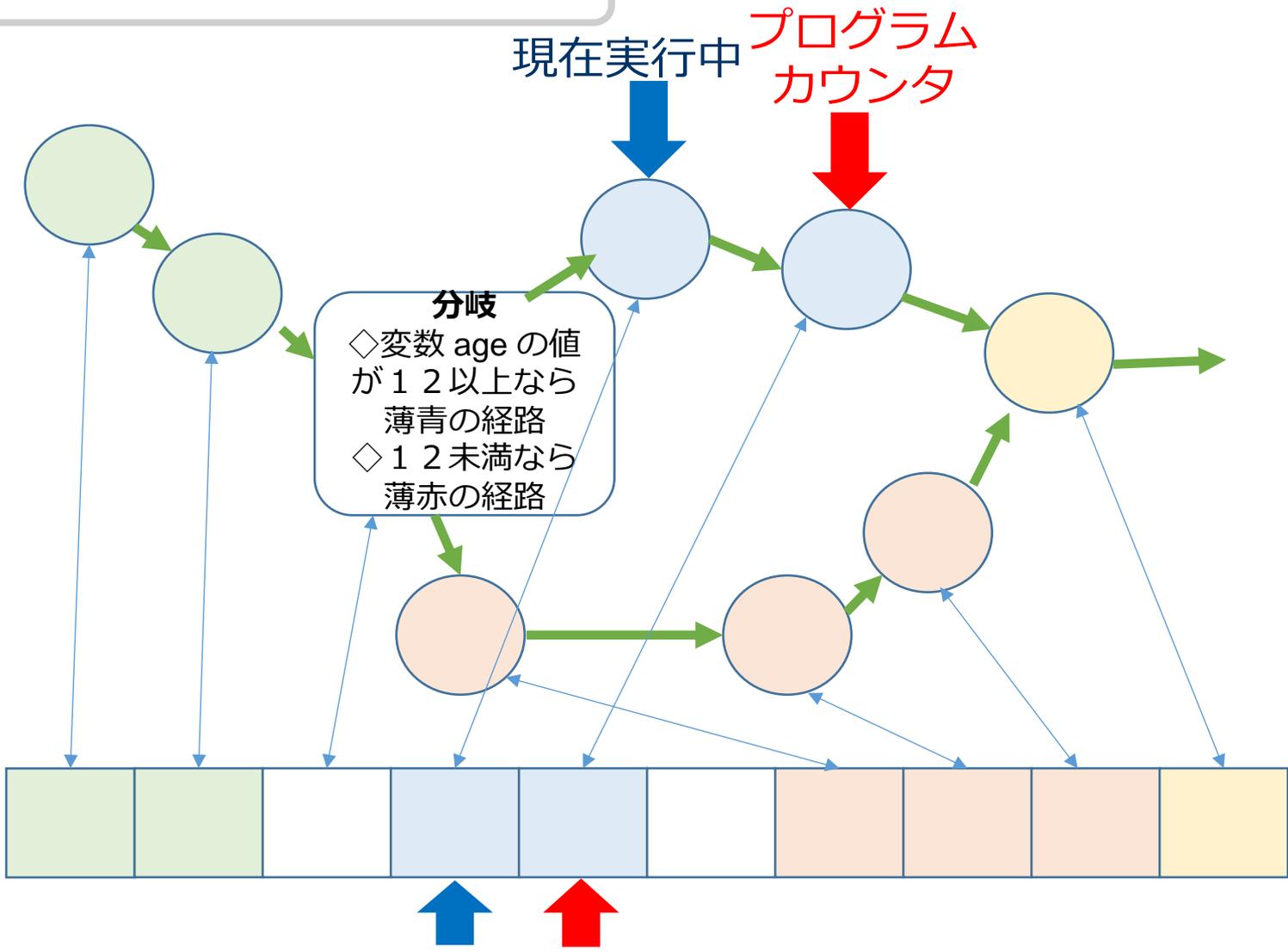
プログラム
カウンタ



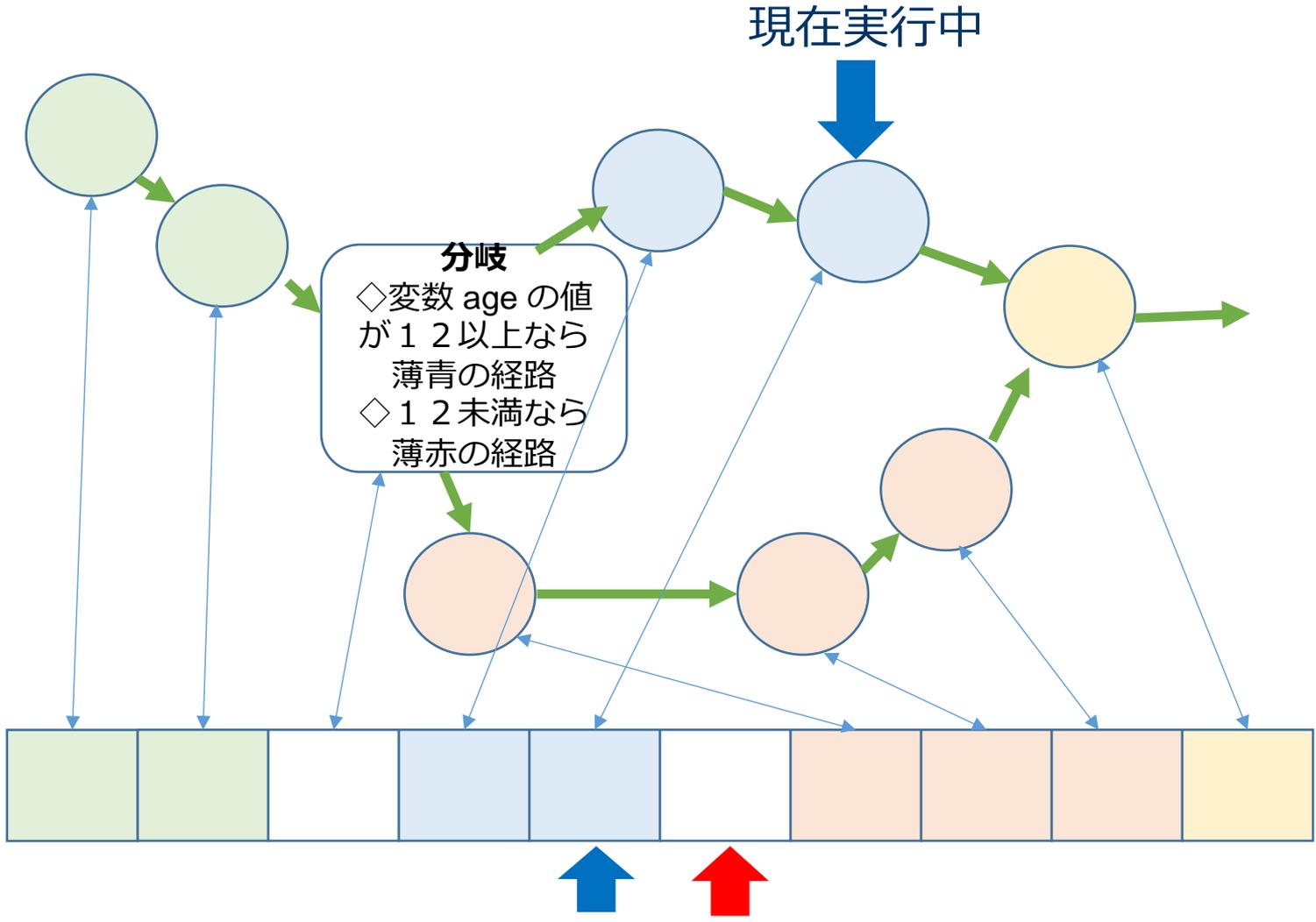




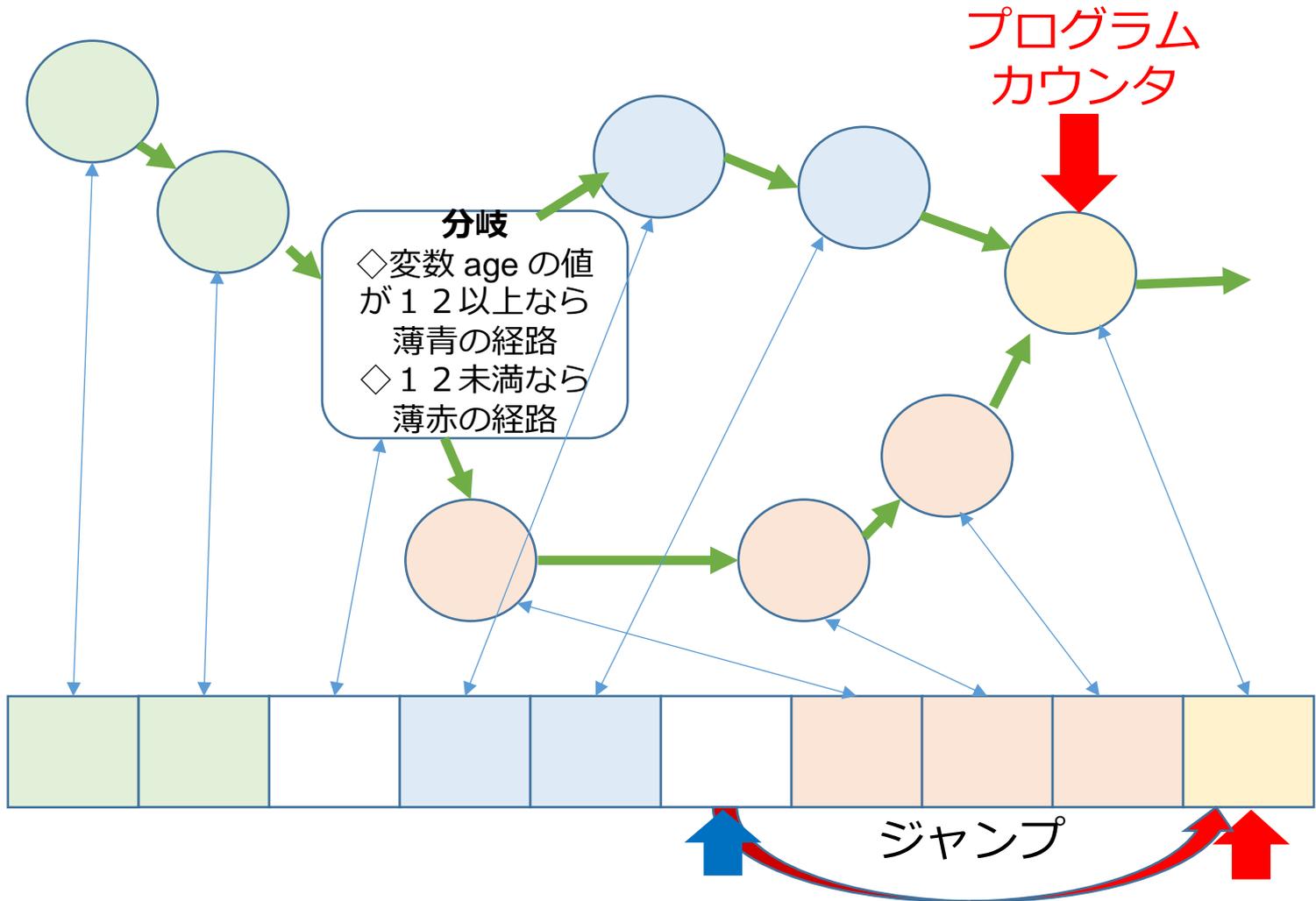
age の値が 1 2 以上 のとき



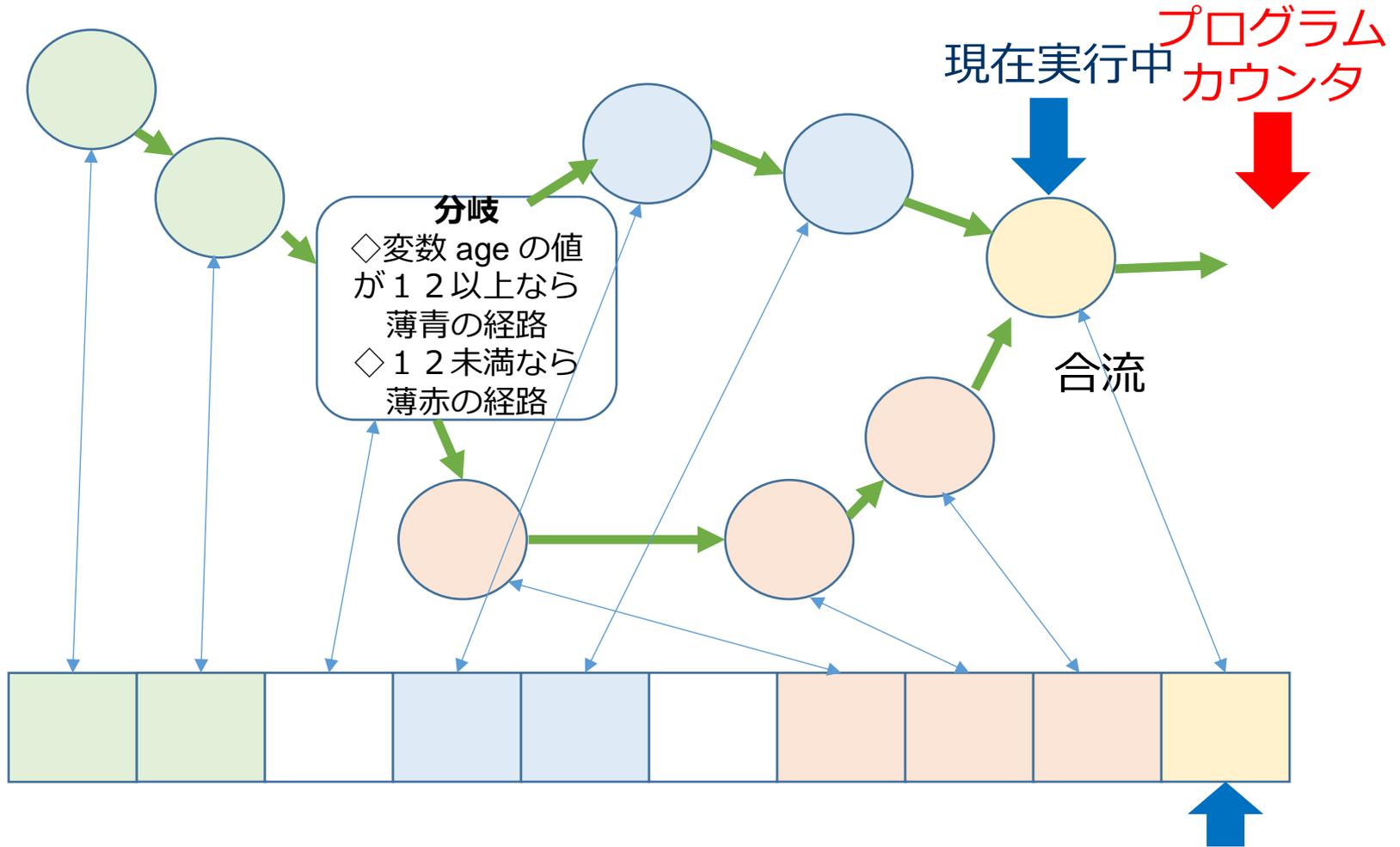
age の値が 1 2 以上 のとき



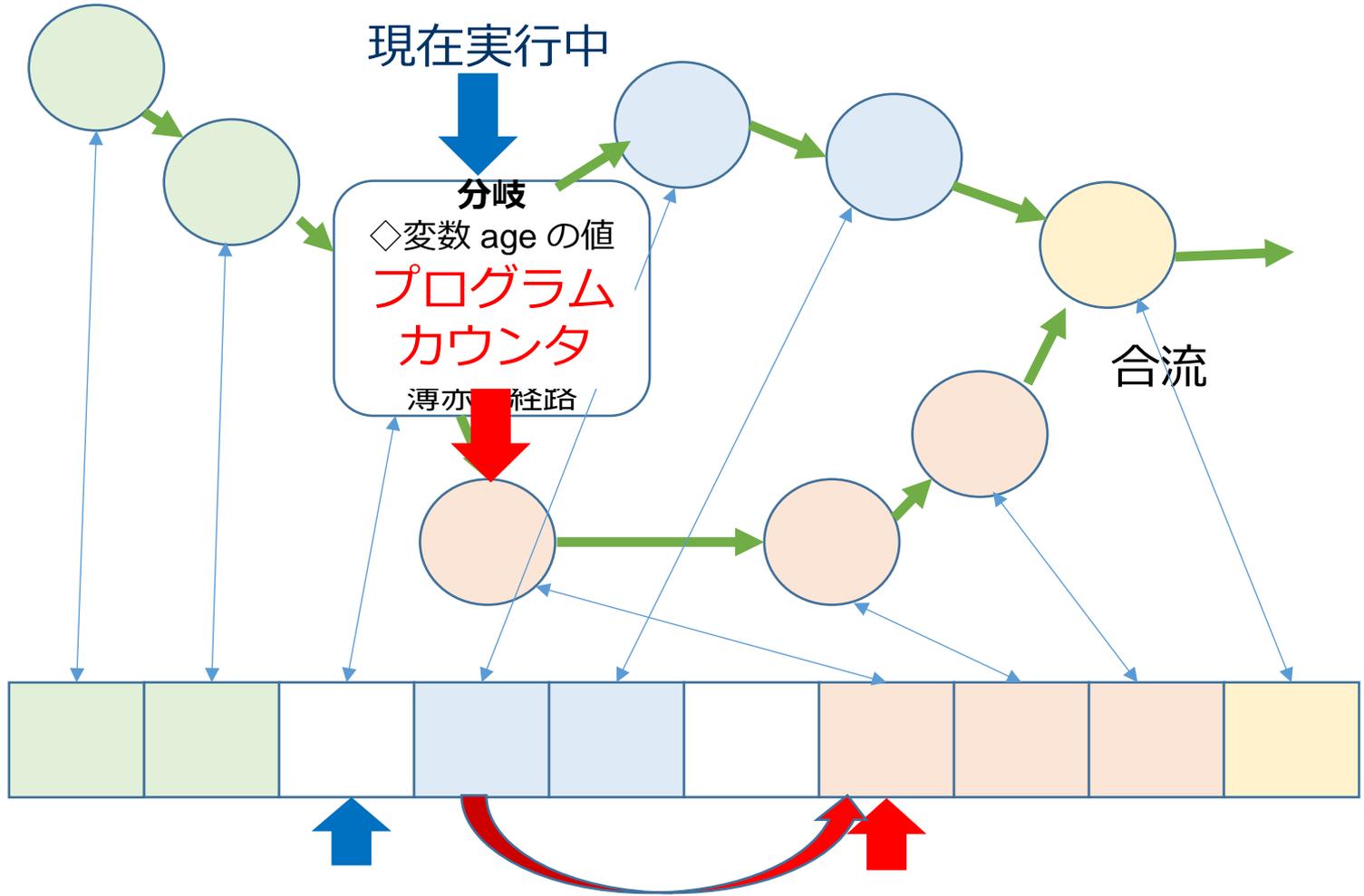
age の値が 1 2 以上のとき



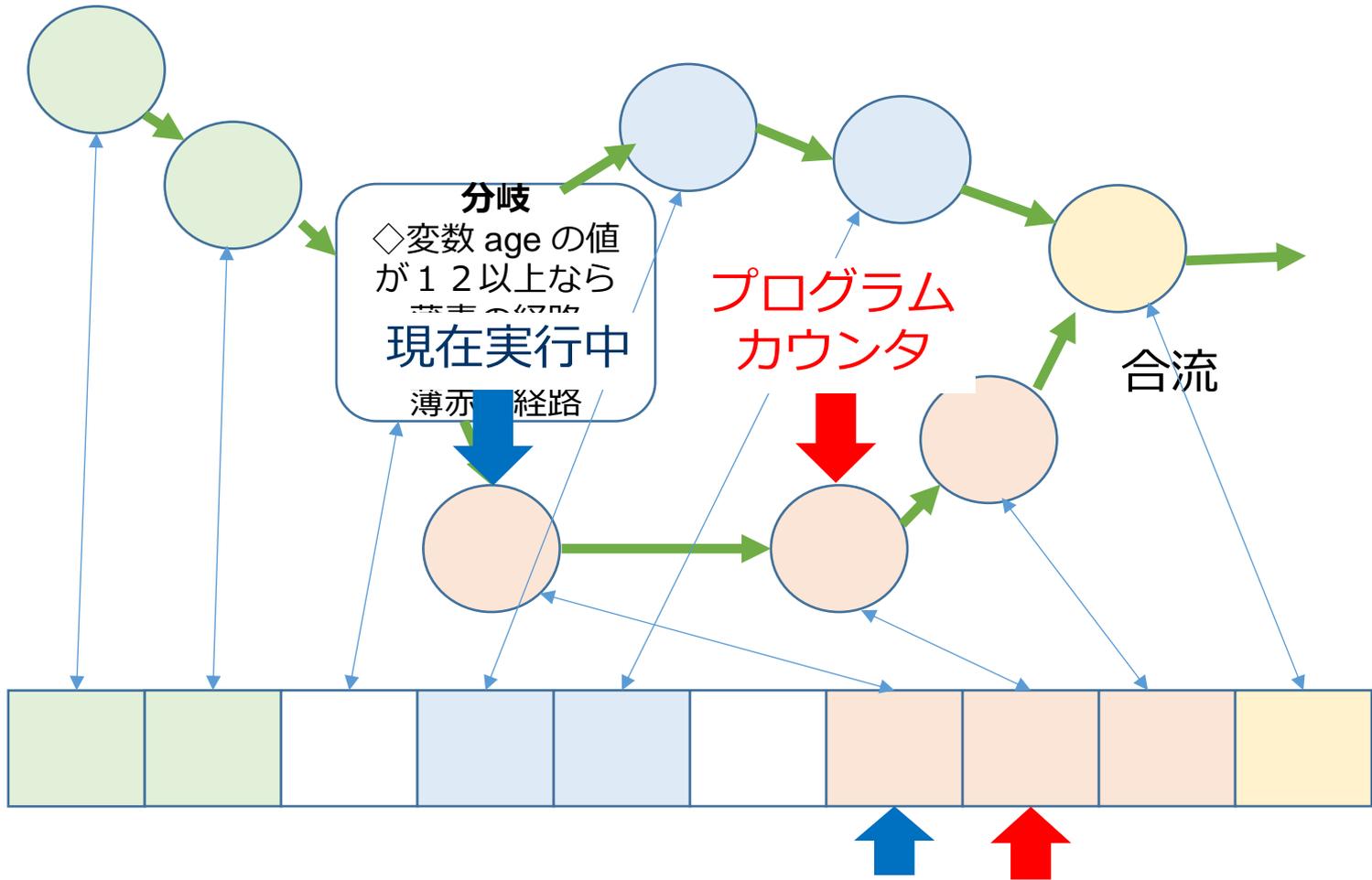
age の値が 1 2 以上 のとき



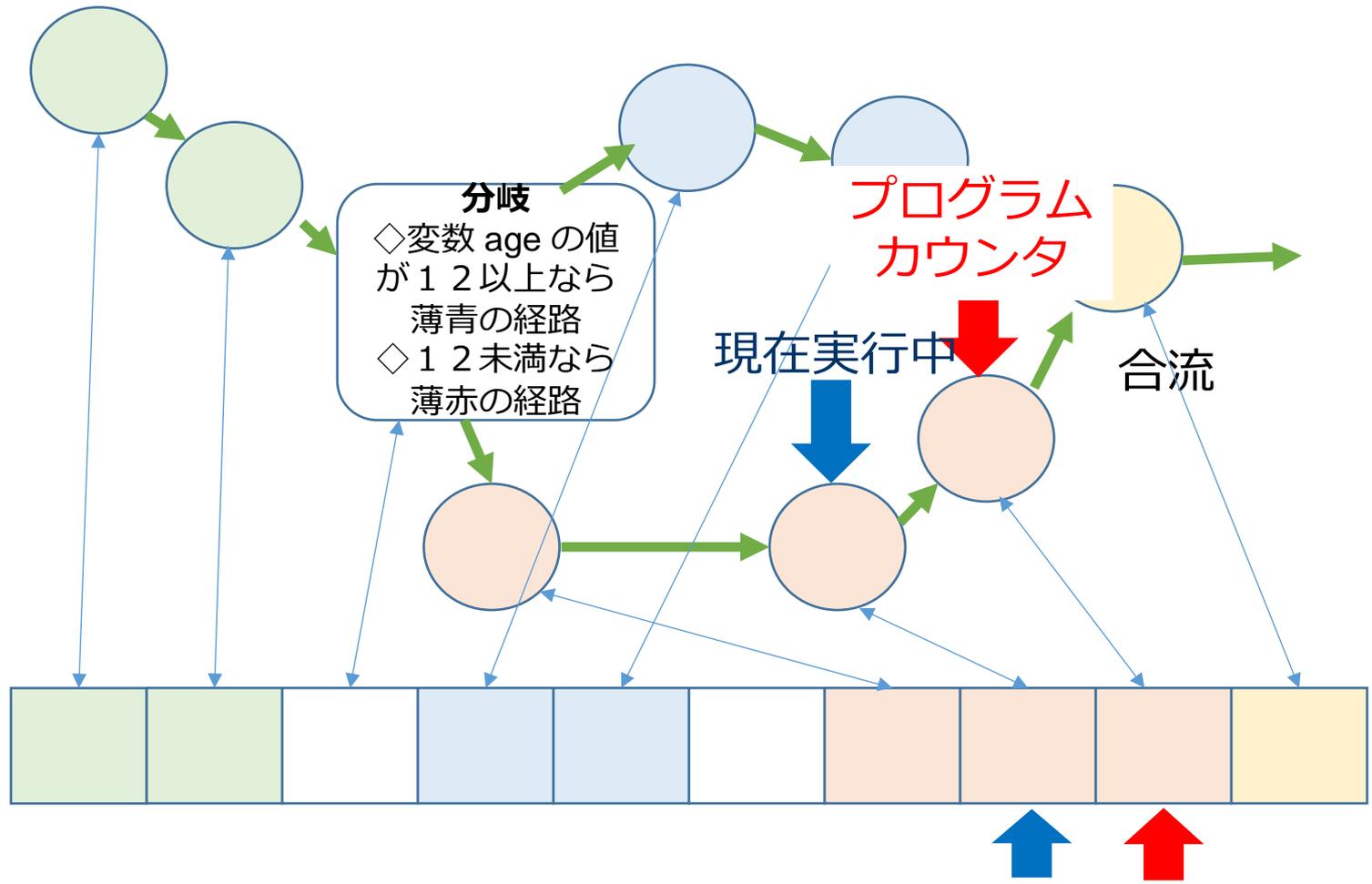
age の値が 1 2 未満 のとき



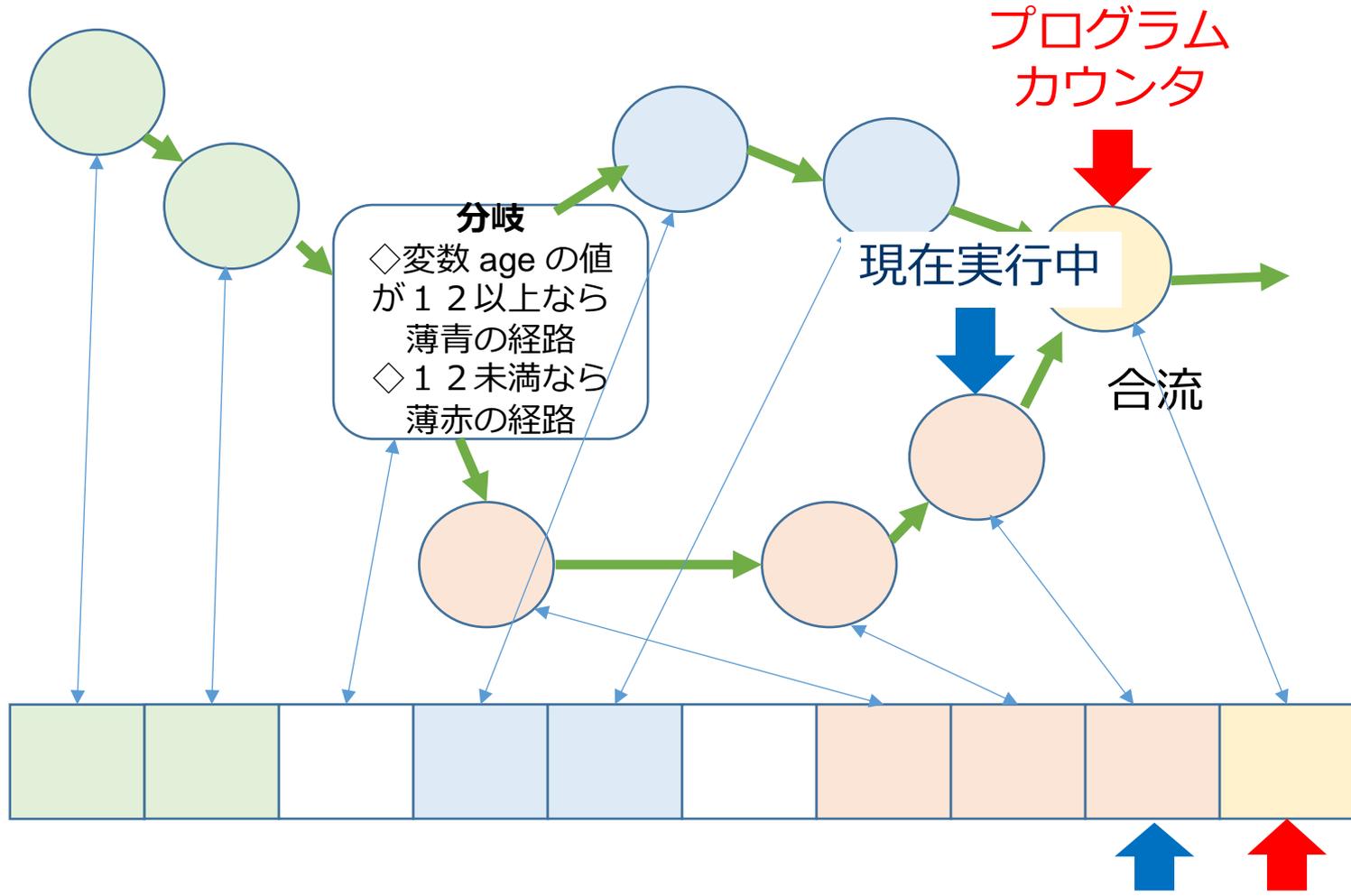
age の値が 1 2 未満 のとき



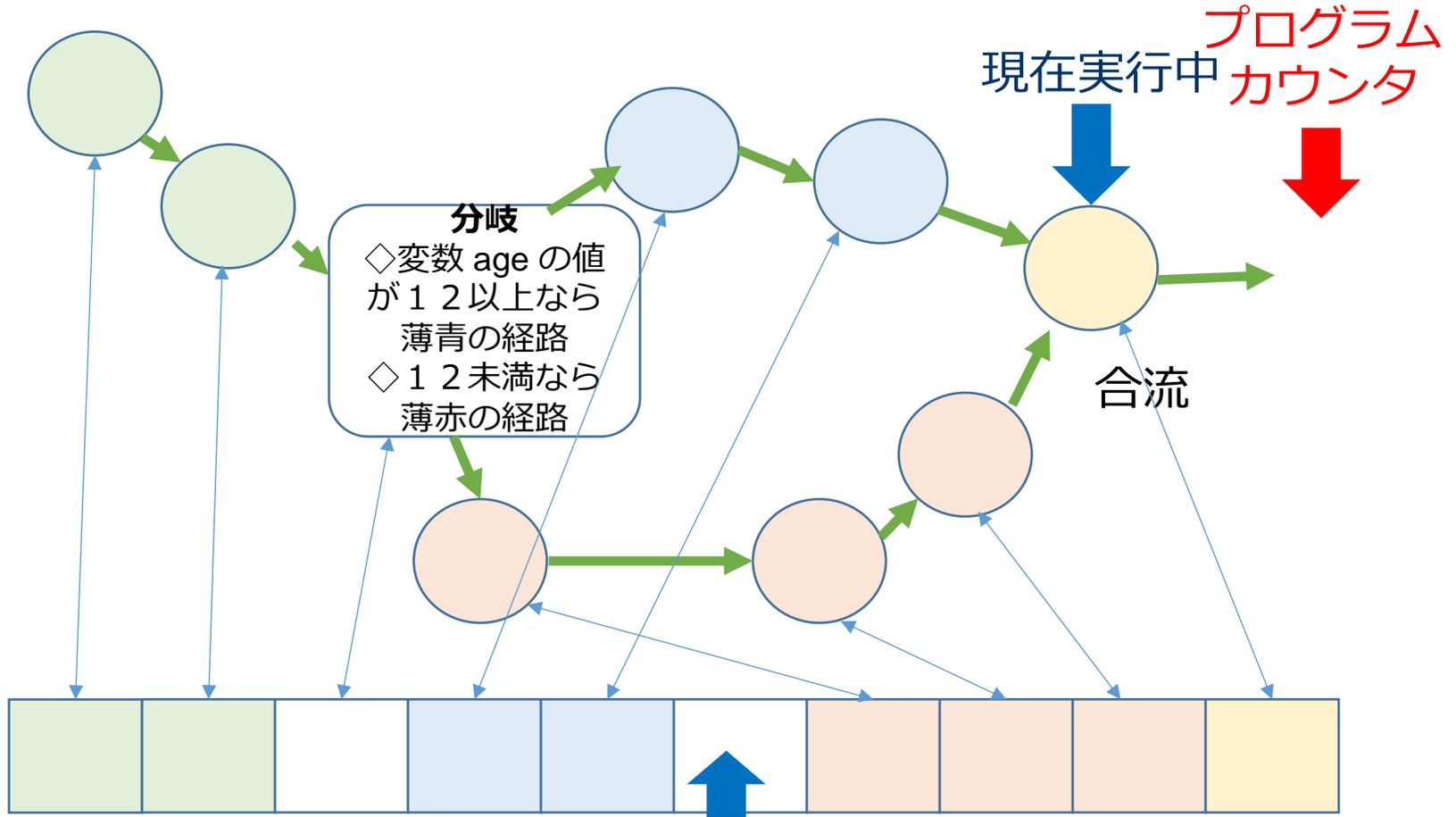
age の値が 1 2 未満 のとき



age の値が 1 2 未満 のとき



age の値が 1 2 未満 のとき



6-2 Visual Studio で プログラムカウンタの表示

プログラムカウンタの表示操作



```
ConsoleApplication1.cpp
(グローバル スコープ)
// ConsoleApplication1.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
//
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
    static int x[8] = { 3, 5, 2, 1, 2, 2 };
    static int y[8];
    int i;
    for (i = 0; i < 8; i++) {
        y[i] = x[i] * 12;
    }

    return 0;
}
```



デバッガを起動済みで、プログラムの実行が中断しているときに・・・

① 「デバッグ」
→ 「ウィンドウ」 → 「レジスタ」

② レジスタが表示される

```
EAX = CCCCCCCC EBX = 7EFDE000 ECX = 00000000 EDX = 00000001 ESI = 00000000 EDI = 001BF424 EIP = 00C3139E
ESP = 001BF958 EBP = 001BF424 EFL = 00000200
0x00c38130 = 00000000
```

- Visual Studio を起動しなさい
- Visual Studio で, Win32 コンソールアプリケーションプロジェクトを新規作成しなさい

プロジェクトの「名前」は何でもよい

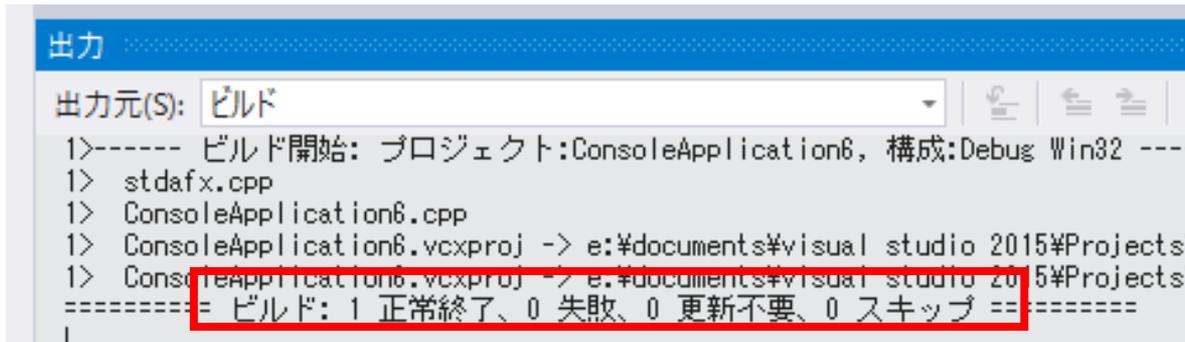
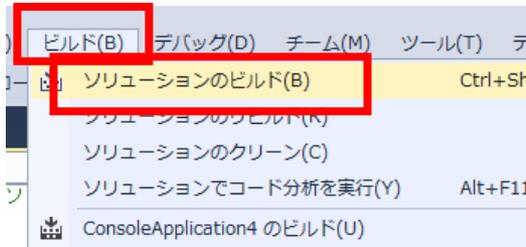
- Visual Studioのエディタを使って、ソースファイルを編集しなさい

```
int main()  
{  
    static int age, p;  
    age = 20;  
    if (age > 12)  
        p = 1800;  
    else  
        p = 500;  
    return 0;  
}
```

追加

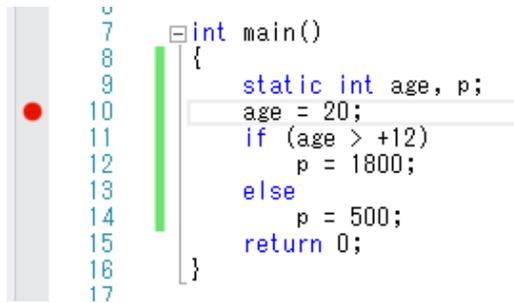
- ビルドしなさい。ビルドのあと「1 正常終了, 0 失敗」の表示を確認しなさい

→ 表示されなければ, プログラムのミスを自分で確認し, 修正して, ビルドをやり直す



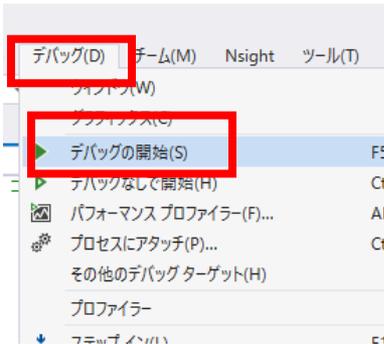
- Visual Studioで「**age = 20;**」の行に、**ブレークポイント**を設定しなさい

```
7 int main()
8 {
9     static int age, p;
10    age = 20;
11    if (age > +12)
12        p = 1800;
13    else
14        p = 500;
15    return 0;
16 }
17
```



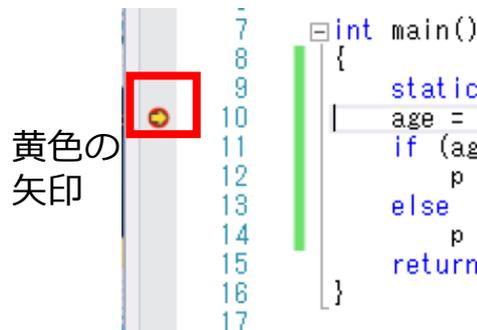
赤丸がブレークポイント
の印

- Visual Studioで、デバッガーを起動しなさい。



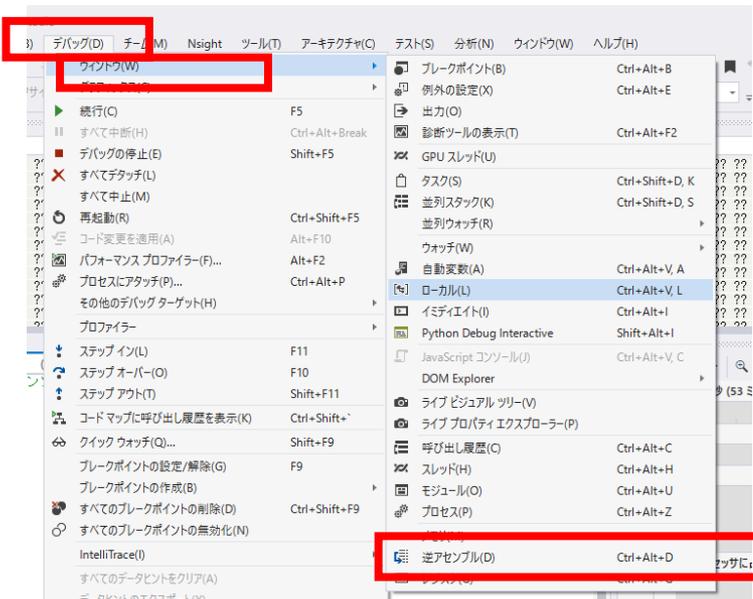
「デバッグ」
→ 「デバッグ開始」

- 「age = 20;」 の行で、実行が中断することを確認しなさい
- あとで使うので、中断したままにしておくこと



「age = 20;」 の行で実行が
中断している

- 「age = 20;」 の行で，実行が中断した状態で，逆アセンブルを行いなさい。



```
アドレス(A): main(void)
表示オプション

int main()
{
01011850 push    ebp
01011851 mov     ebp,esp
01011853 sub     esp,0C0h
01011859 push    ebx
0101185A push    esi
0101185B push    edi
0101185C lea    edi,[ebp-0C0h]
01011862 mov     ecx,30h
01011867 mov     eax,0CCCCCCCCh
0101186C rep stos dword ptr es:[edi]
    static int age, p;
    age = 20;
0101186E mov     dword ptr [age (01019138h)],14h
    if (age > +12)
01011878 cmp     dword ptr [age (01019138h)],0Ch
0101187F jle     main+3Dh (0101188Dh)
    p = 1800;
01011881 mov     dword ptr [p (0101913Ch)],708h
    else
01011888 jmp     main+47h (01011897h)
    p = 500;
0101188D mov     dword ptr [p (0101913Ch)],1F4h
    return 0;
01011897 xor     eax,eax
}
01011899 pop     edi
0101189A pop     esi
```

① 「デバッグ」 → 「ウインドウ」 → 「逆アセンブル」

② 逆アセンブルの結果が表示される

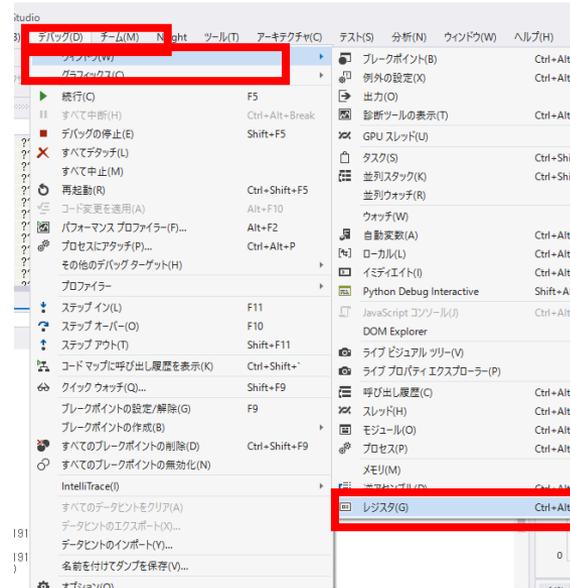
- 「age = 20;」 の行で，実行が中断した状態で，レジスタの中身を表示させなさい。手順は次の通り。

EIP はプログラムカウンタ

```

01 1186E mov     dword ptr [age (01019138h)],14h
    if (age > +12)
01011878 cmp     dword ptr [age (01019138h)],0Ch
0101187F jle     main+3Dh (0101188Dh)
    p = 1800;
01011881 mov     dword ptr [p (0101913Ch)],708h
    else
01011888 jmp     main+47h (01011897h)
    p = 500;
0101188D mov     dword ptr [p (0101913Ch)],1F4h
    return 0;
01011897 xor     eax,eax
  
```

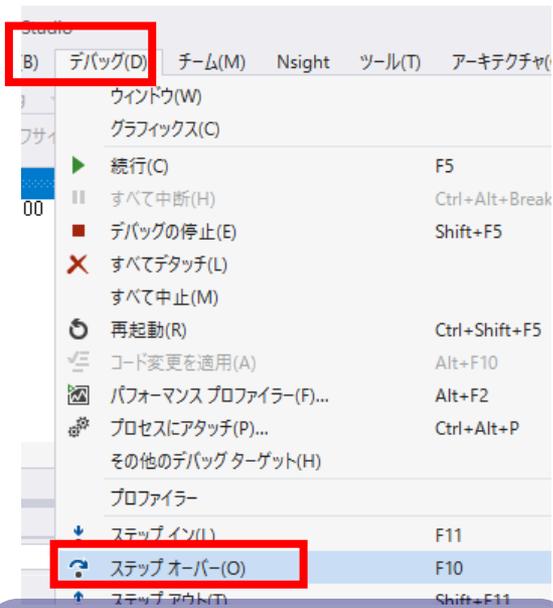
デバッガーを起動済みで，プログラムの実行が中断しているときに・・・



② レジスタが表示される。EIP に注目

① 「デバッグ」
→ 「ウィンドウ」 → 「レジスタ」

- ステップオーバーの操作を 1 回ずつ行いながら、レジスタウインドウの中の EIP の変化を確認しなさい。



「デバッグ」
→ 「ステップオーバー」
(あるいは F10 キー)

```
age = 20;
0101168E mov     dword ptr [age (01019138h)],14h
        if (age > +12)
01011678 cmp     dword ptr [age (01019138h)],0Ch
0101167F jle     main+30h (0101168Dh)
        p = 1800;
01011681 mov     dword ptr [p (0101913Ch)],708h
        else
0101168B jmp     main+47h (01011697h)
        p = 500;
0101168D mov     dword ptr [p (0101913Ch)],1F4h
        return 0;
01011697 xor     eax,eax
        \
```

アセンブリ言語の中で：

赤丸： **ブレークポイント**

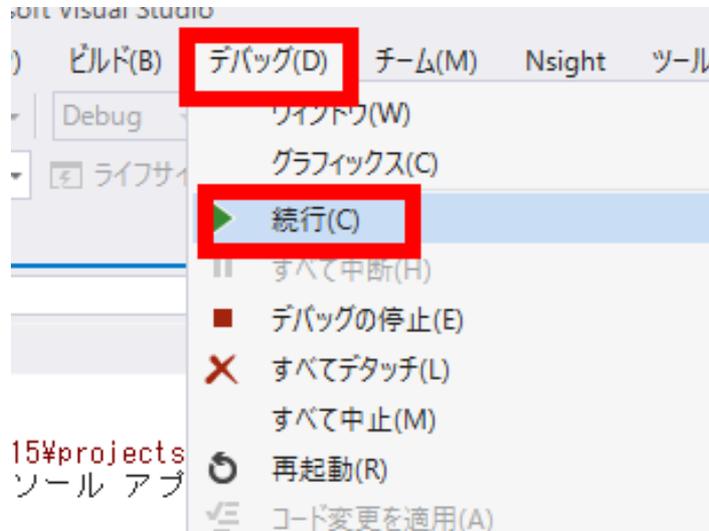
黄色矢印： **プログラムカウンタ**

レジスタ

EAX = CCCCCCCC EBX = 013C5000 ECX = 00000000 EDX = 01019590 ESI = 01011041 EDI = 0111FAB4 **EIP = 0101168E** ESP = 0111F9E8 EBP = 0111FAB4 EFL = 00000200
0x01019138 = 00000000

EIP は
プログラムカウンタ

- 最後に、プログラム実行の再開の操作を行いなさい。これで、デバッガーが終了する。



「デバッグ」
→ 「続行」

演習



- 「age = 20;」 の行を 「age = 10;」 に変えて, 今の手順を繰り返さない.
- ジャンプの様子が変わるので確認しない

演習



- 次のプログラムでプログラムカウンタの値の変化の様子を確認しなさい。

```
int main()
{
    static int i, s;
    s = 0;
    for (i = 1; i <= 3; i++)
        s = s + (i * i);
    return 0;
}
```

5-3 命令実行サイクル

Visual C++ 言語とアセンブリ言語



Visual C++ の
プログラム

```
x = 3;  
y = 4;  
z = x + y;
```

アセンブリ言語

```
mov     dword ptr [x (0CA9138h)],3  
mov     dword ptr [y (0CA913Ch)],4  
mov     eax,dword ptr [x (0CA9138h)]  
add     eax,dword ptr [y (0CA913Ch)]  
mov     dword ptr [z (0CA9140h)],eax
```

いまから、この3行を図解で説明

プロセッサの仕組みの概要

