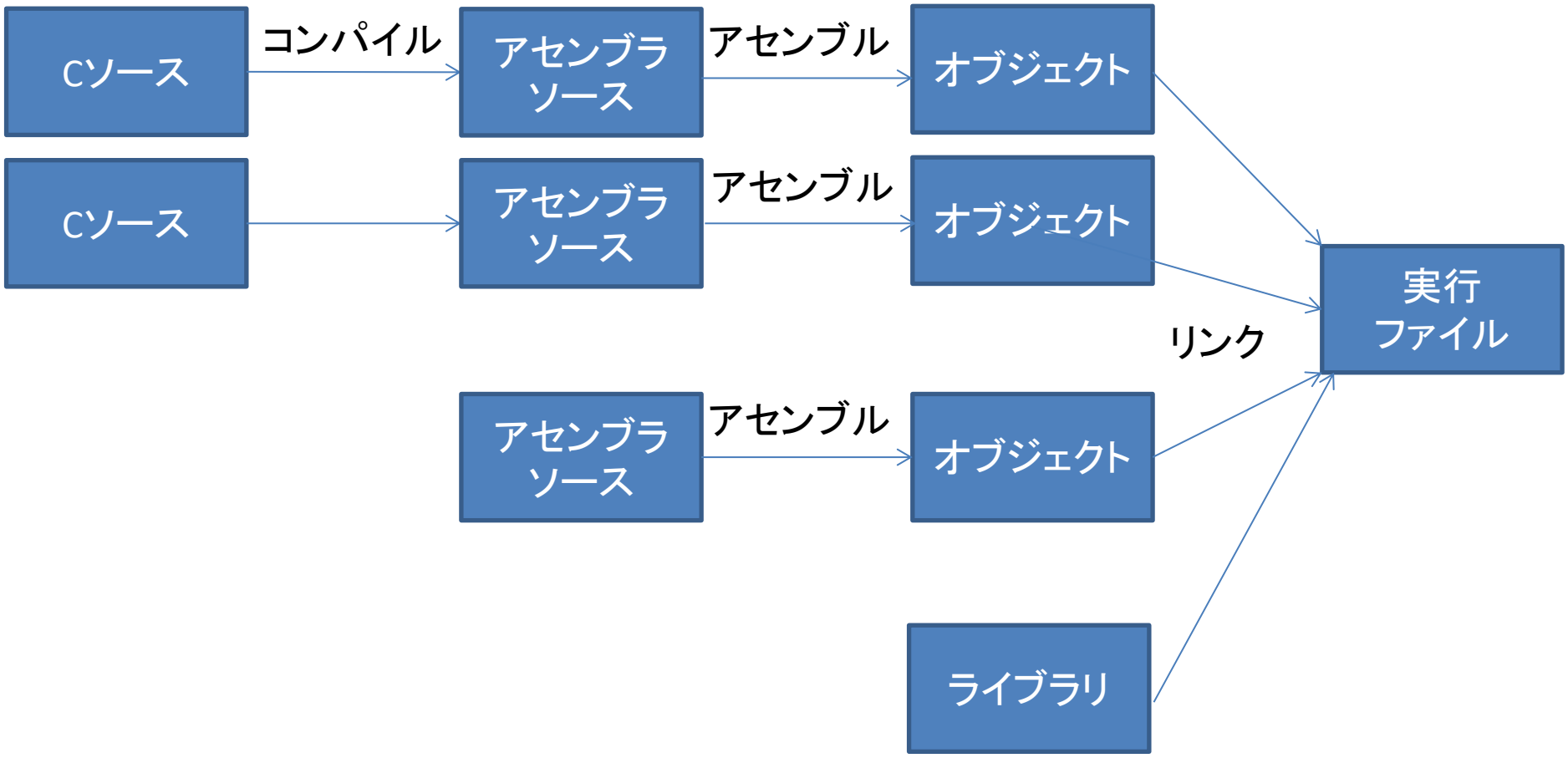
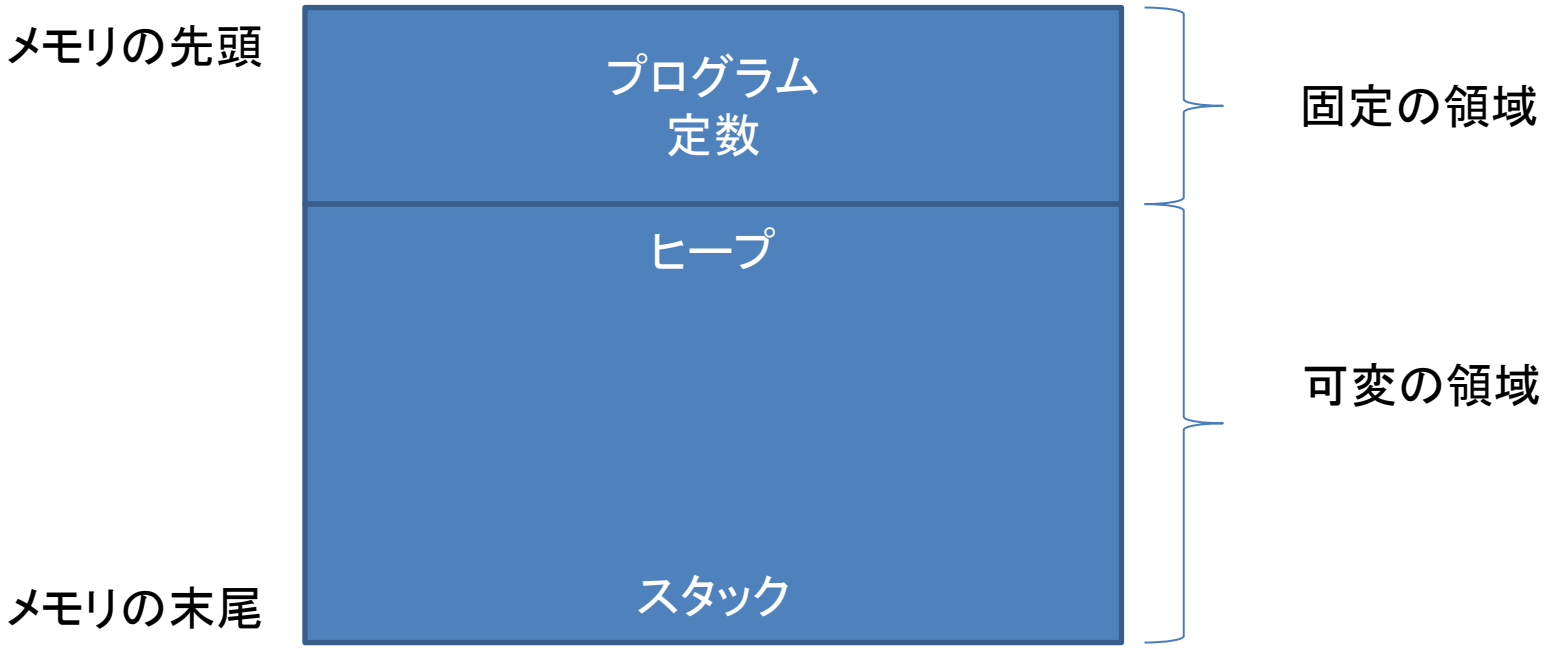
C言語での実行ファイルの生成



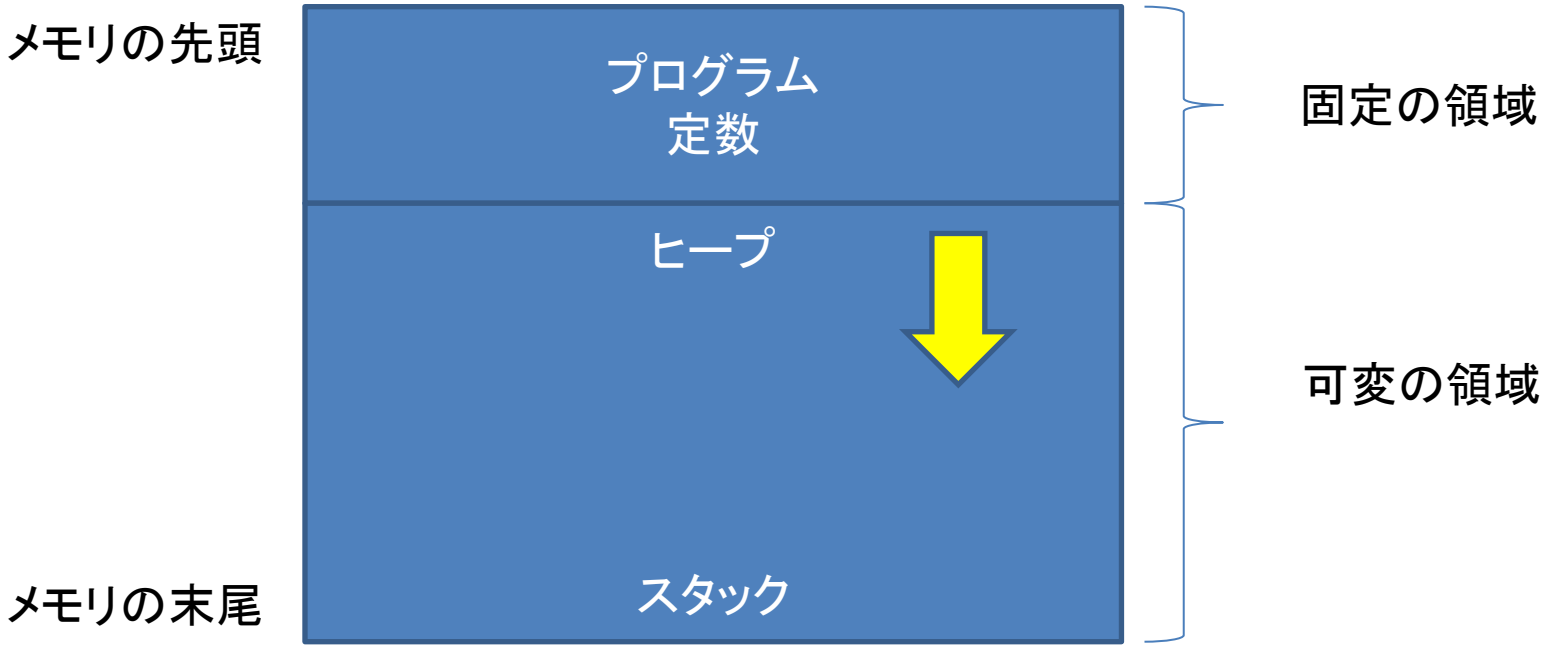
実行イメージ

OSから実行するとおおまかには以下のように配置される。
ハードウェア/OSの仮想メモリ機能で他のプログラムの領域は見えない。



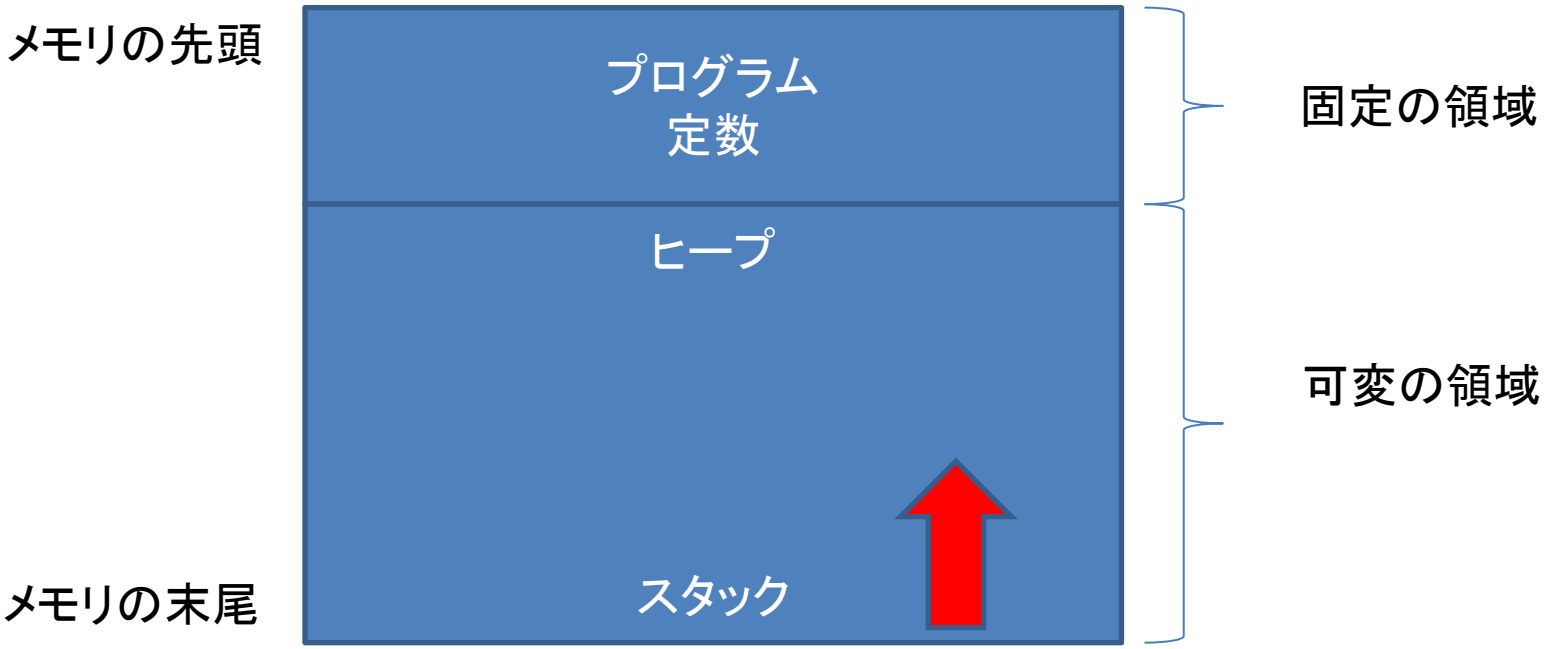
ヒープ

グローバル変数、static 変数が配置される。
malloc で動的に確保したメモリは、ヒープから動的に確保され、
アドレスが小さいほうから大きな方向に成長。freeで取られた領域は解放される。(freeを呼ばないと解放されない)



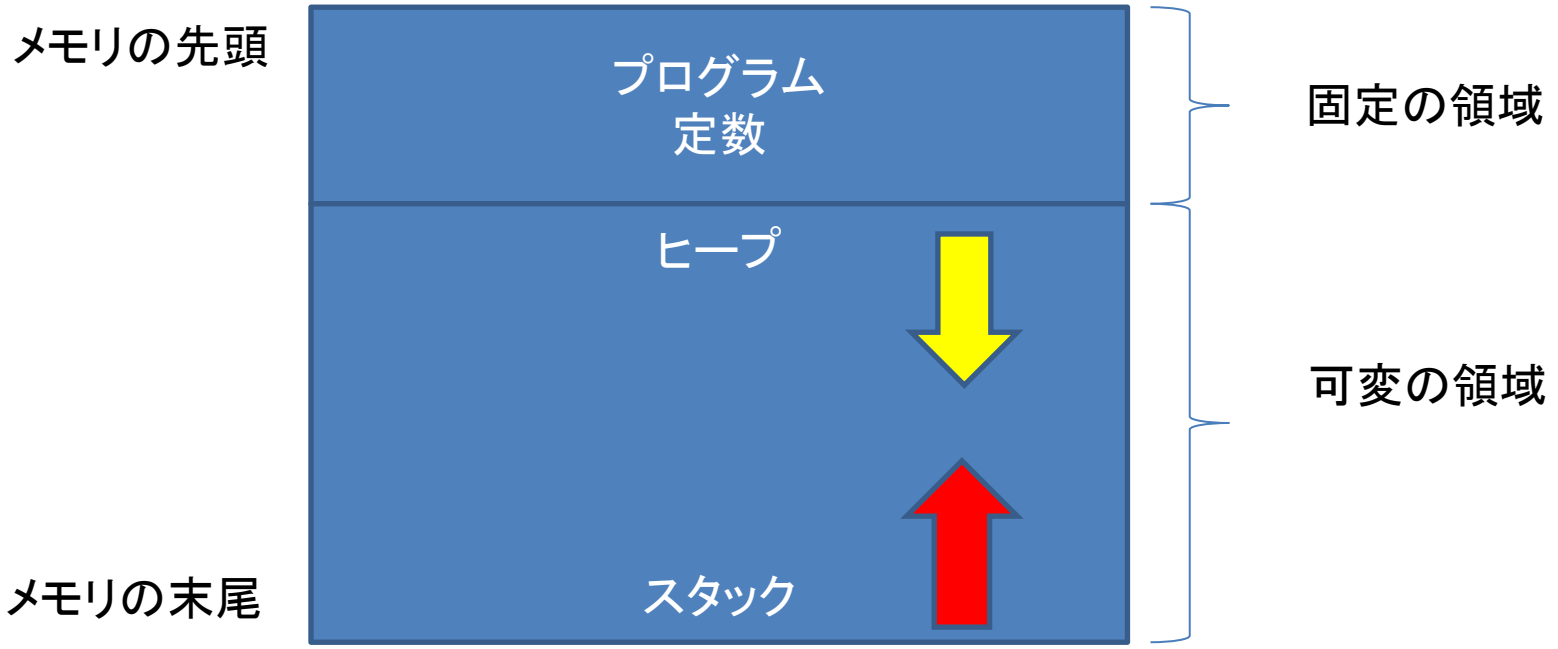
スタック

サブルーチンを呼ぶと、戻り先がスタックに積まれる。
ローカル変数はスタック上に確保される。
アドレスの大きいほうから小さいほうに成長。
サブルーチンからリターンするとスタックは呼ばれる前の状態に戻る。
(実際には、Cコンパイラがそういうコードを生成する)



ヒープとスタックの危険な関係

大きなヒープを取るとスタック(=戻り先やローカル変数)を上書きすることもあるし、
巨大なローカル変数を取るとヒープ(=グローバル変数やmallocしたメモリ)を
上書きすることも起こりえる。



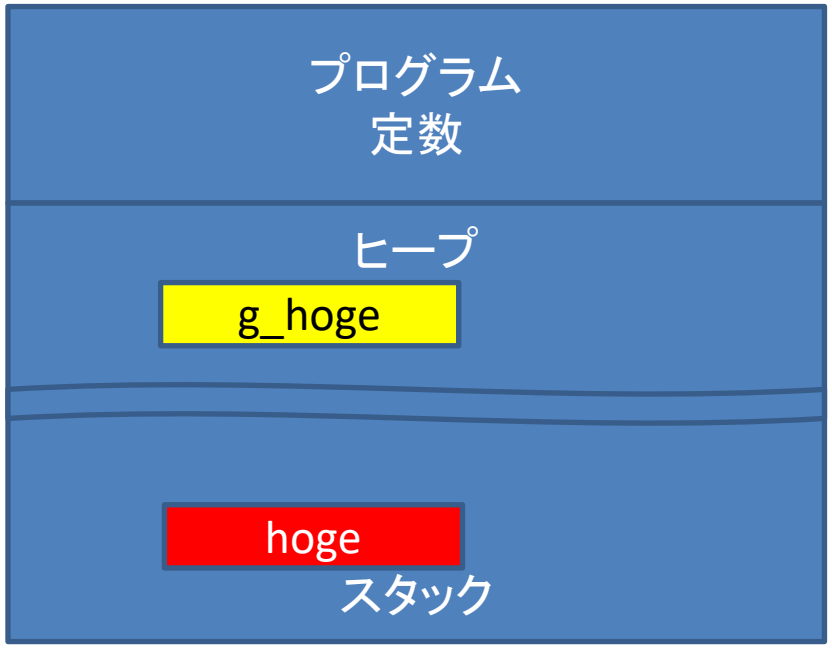
変数、構造体の格納イメージ

メモリの先頭

```
int g_hoge;

void func()
{
  int hoge;
}
```

メモリの末尾

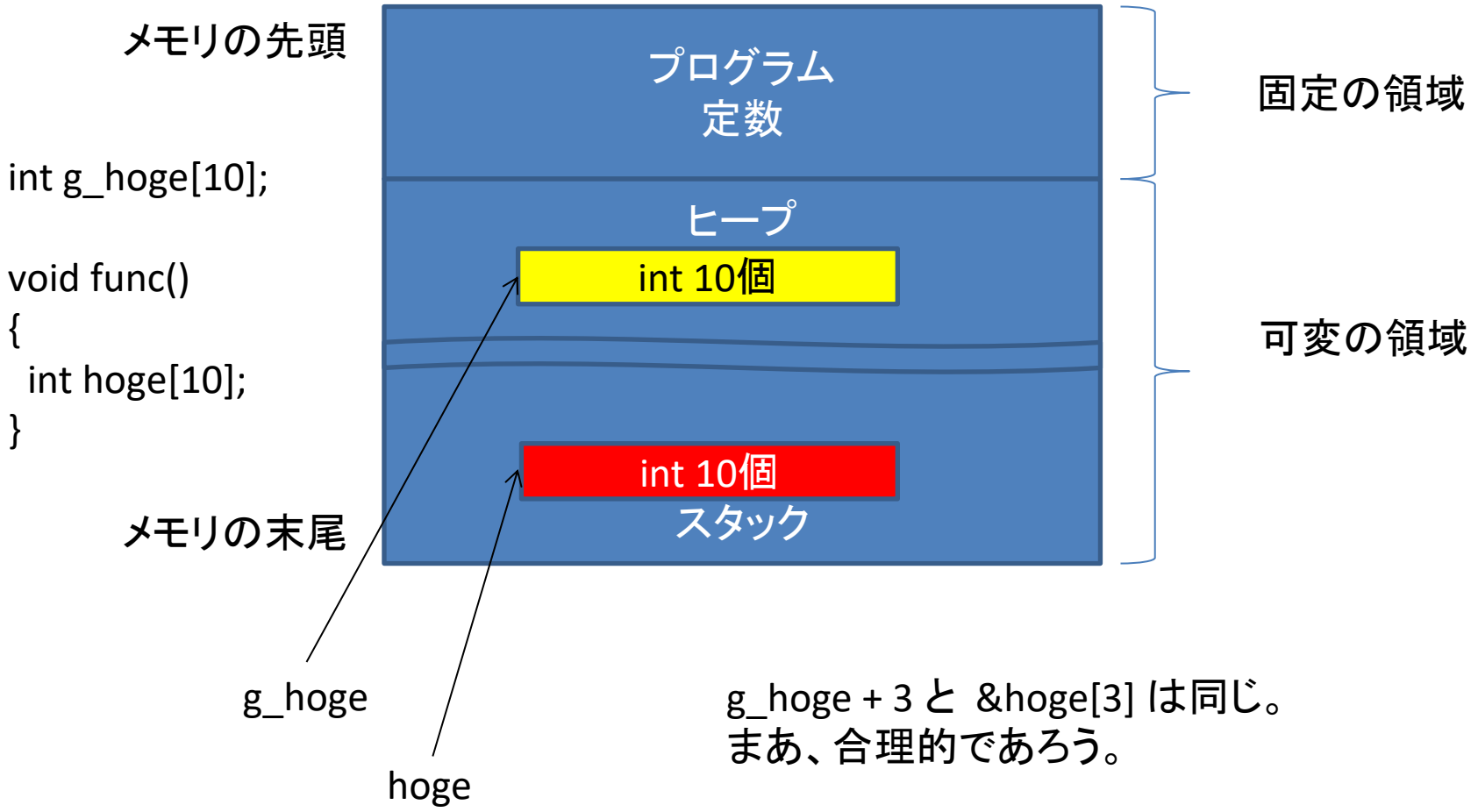


固定の領域

可変の領域

もし、アドレスを参照したい場合があれば、それぞれ、&g_hoge、&hoge とする。

配列の格納イメージ


メモリの先頭
int g_hoge[10];
void func()
{
int hoge[10];
}
メモリの末尾
プログラム
定数
ヒープ
int 10個
int 10個
スタック
固定の領域
可変の領域
g_hoge
hoge
g_hoge + 3 と &hoge[3] は同じ。
まあ、合理的であろう。