

問2 コンパイラの最適化に関する次の記述を読んで、設問1～3に答えよ。

コンパイラとは、プログラム言語で記述された原始プログラムを翻訳して目的プログラムを生成するためのソフトウェアである。コンパイラの機能の一つに最適化がある。最適化では、原始プログラムを翻訳する過程で、プログラムの実行時間を短くするために原始プログラムの構造を変換する。最適化の方法の例を表1に示す。

表1 最適化の方法の例

最適化の方法	内容
関数のインライン展開	関数を呼び出す箇所に、呼び出される関数のプログラムを展開する。
共通部分式の削除	同じ式が複数の箇所に存在し、それらの式で使用している変数の値が変更されず、式の値が変化しないとき、その式の値を作業用変数に格納する文を追加し、複数の同じ式をその作業用変数で置き換える。
定数の畳込み	プログラム中の定数同士の計算式を、その計算結果で置き換える。
定数伝播	変数を定数で置き換える。
無用命令の削除	プログラムの実行結果に影響しない文を削除する。
ループ内不変式の移動	ループ中で値の変化しない式があるとき、その式をループの外に移動する。
ループのアンローリング	ループ中の繰返しの処理を展開する。

擬似言語の形式で記述したプログラムの一部（以下、プログラム1という）に対して、表1の最適化の方法を複数組み合わせ最適化した例を、表2に示す。

[プログラム1]

```

■ i: 0, i ≤ 1, 1
  |
  | • x[i] ← y[i] + m + n
  | • v[i] ← w[i] + m + n
  |
■
    
```

表2 プログラム1の最適化の例

最適化の方法	プログラム1の変換
①	<pre> ■ i: 0, i ≤ 1, 1       • t ← m + n   • x[i] ← y[i] + t   • v[i] ← w[i] + t     ■                     </pre>
②	<pre>   • t ← m + n   ■ i: 0, i ≤ 1, 1       • x[i] ← y[i] + t   • v[i] ← w[i] + t     ■                     </pre>
③	<pre>   • t ← m + n   • x[0] ← y[0] + t   • v[0] ← w[0] + t   • x[1] ← y[1] + t   • v[1] ← w[1] + t   • i ← 2                     </pre>

設問1 次の記述中の  に入れる正しい答えを、解答群の中から選べ。

表2において、最適化の方法を①, ②, ③の順で適用するとき、②で適用される最適化の方法は  であり、③で適用される最適化の方法は  である。

a, bに関する解答群

- |               |              |
|---------------|--------------|
| ア 関数のインライン展開  | イ 共通部分式の削除   |
| ウ 定数の畳込み      | エ 定数伝播       |
| オ 無用命令の削除     | カ ループ内不変式の移動 |
| キ ループのアンローリング |              |

設問2 次の記述中の          に入れる正しい答えを、解答群の中から選べ。

関数のインライン展開の例を図1に示す。図1の関数 func をインライン展開した場合、実引数 10 の値が仮引数 p に代入され、その値 10 が関数 func 内の変数 p の値になる (図1①)。次に、         c の方法を適用することによって、条件式  $p \geq 10$  の判定結果や p を含んだ計算式の計算結果がコンパイル時に確定する (図1②)。その結果、func(10) を値 15 に置き換えることができる (図1③)。

なお、#w はコンパイラが最適化したときに生成した整数型の変数である。

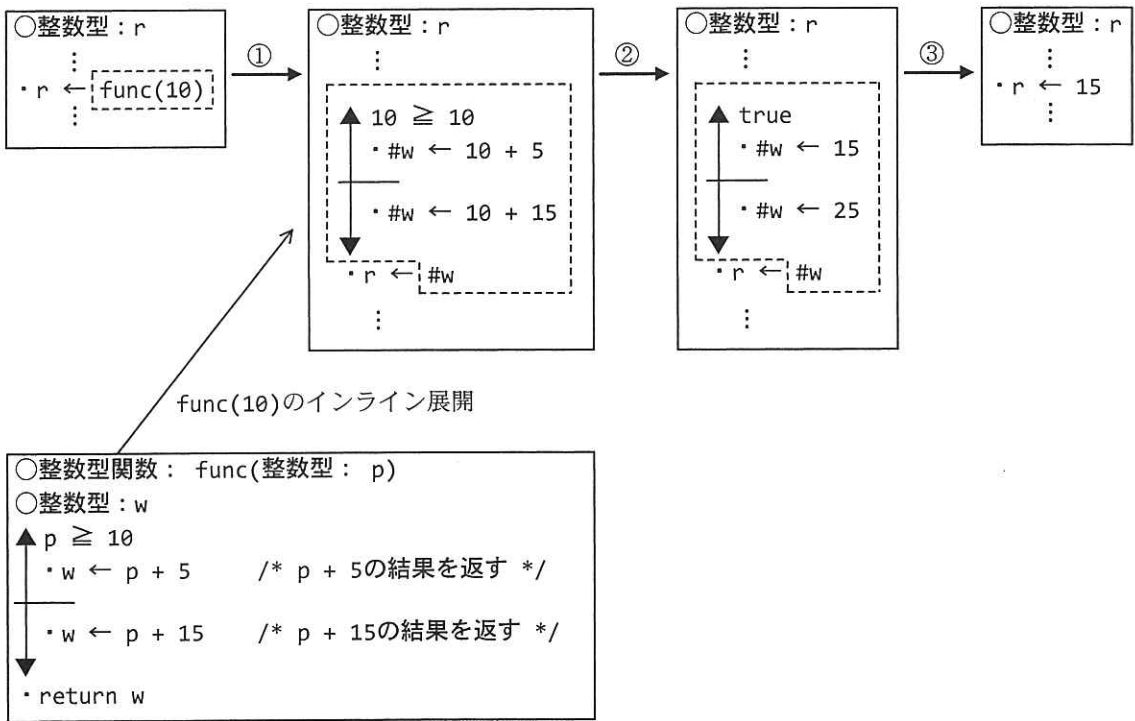


図1 関数のインライン展開の例

cに関する解答群

- |              |               |
|--------------|---------------|
| ア 共通部分式の削除   | イ 定数の畳込み      |
| ウ 定数伝播       | エ 無用命令の削除     |
| オ ループ内不変式の移動 | カ ループのアンローリング |

設問 3 次の記述中の  に入れる適切な答えを、解答群の中から選べ。

最適化をしないときと最適化をしたときとで浮動小数点数の演算の結果が異なる場合がある。プログラム 1 に対して表 2 の最適化をしなかったものと、最適化をしたものとの、 $y[0]=307000000.0$ ,  $y[1]=305000000.0$ ,  $m=-303000000.0$ ,  $n=4.0$  を与えて実行した。その結果の  $x[0]$ ,  $x[1]$  が次のとおりになった。ここで、同一優先順位の算術演算子は、左から順に演算する。

最適化をしないとき：  $x[0]=4000004.0$ ,  $x[1]=2000004.0$

最適化をしたとき：  $x[0]=4000000.0$ ,  $x[1]=2000000.0$

この実行結果が異なる原因としては、 d  の方法の適用によって演算順序が変化したことで、 e  が発生したからである。ここで、浮動小数点数の演算は単精度（仮数部は 23 ビット）で計算しているものとする。

dに関する解答群

- |               |              |
|---------------|--------------|
| ア 関数のインライン展開  | イ 共通部分式の削除   |
| ウ 定数の畳込み      | エ 定数伝播       |
| オ 無用命令の削除     | カ ループ内不変式の移動 |
| キ ループのアンローリング |              |

eに関する解答群

- |        |       |        |        |
|--------|-------|--------|--------|
| ア 桁あふれ | イ 桁落ち | ウ 情報落ち | エ 丸め誤差 |
|--------|-------|--------|--------|