

CodeWarrior®

Error Reference



CodeWarrior は出荷直前でも改良されることがあるので、このマニュアルに記載されている内容の一部が本物のソフトウェアの動きと異なることがあるかもしれません。最新の情報については CodeWarrior の「Release Notes」フォルダをご覧ください。

Revised: 980829-JDR-JP990127

Metrowerks CodeWarrior © Copyright 1993-1998 by Metrowerks Inc. and its Licensors. All rights reserved.

お客様は、本 CD に記録されている文書を個人使用目的に限り、プリントすることができます。この場合を除いて、Metrowerks Inc. からの書面による承諾なしに、本 CD に記録されている文書の全部、または、一部をいかなる形態、方法（電子的、物理的な複製、または、写真複写、録音録画、その他すべての情報記録、再生システムを含む）により、複製または伝達することを禁じます。

Metrowerks の名称、ロゴ、CodeWarrior、Software at Work は、Metrowerks Inc. の登録商標です。

PowerPlant、PowerPlant Constructor は、Metrowerks Inc. の商標です。

記載の商標および登録商標は、各社が保有します。

CD に記録されているすべてのソフトウェアおよび文書は、CodeWarrior QuickStart の巻末に記述されているライセンス契約が適用されます。

連絡先：

Japan	メトロワークス株式会社 150-0042 東京都渋谷区宇田川町 36-6 ワールド宇田川ビル 8F TEL : (03) 3780-6091 FAX : (03) 3780-6092
U.S.A.	Metrowerks Corporation 9801 Metric Boulevard, Suite 100 Austin, TX 78758 U.S.A.
Canada	Metrowerks Inc. 1500 du College, Suite 300 Ville St-Laurent, QC Canada H4L 5G6
WWW サーバ	http://www.metrowerks.com
ユーザー登録	j-register@metrowerks.com
テクニカルサポート	j-support@metrowerks.com
購入 / 契約更新	j-sales@metrowerks.com
インフォメーション	j-info@metrowerks.com

目次

第 1 章 紹介	5
Metrowerks の西暦 2000 年問題対応	5
このマニュアルの概要	5
このマニュアルにおける表記方法	5
エラーに影響する環境設定	6
第 2 章 C/C++ コンパイラのエラー	7
C/C++ コンパイラのエラーメッセージ	7
シンボル名 (C/C++)	7
句読記号 (C/C++)	8
A ~ C (C/C++)	13
D ~ F (C/C++)	18
G ~ I (C/C++)	23
J ~ L (C/C++)	47
M ~ O (C/C++)	48
P ~ R (C/C++)	50
S ~ T (C/C++)	55
U ~ Z (C/C++)	60
索引	65



第 1 章 紹介

Metrowerks の西暦 2000 年問題対応

ライセンス契約に基づいて Metrowerks が提供する製品は、ホストまたはターゲットオペレーティングシステムによって提供される日付データを利用して内部プロセスの日付データ（ファイル修正日など）を処理しています。ゆえに、西暦 2000 年問題から生じる製品のオペレーションは、ホストまたはターゲットオペレーティングシステムの西暦 2000 年問題対応によるものです。Microsoft 社、Sun Microsystems 社、Apple Computer 社などの西暦 2000 年問題についての文書をお読みください。Metrowerks 製品自体は日付データを処理しないため、西暦 2000 年問題とは無関係です。

詳細は、<http://www.metrowerks.com/about/y2k.html> をご覧ください。

このマニュアルの概要

このマニュアルでは CodeWarrior コンパイラやリンカの使用中に遭遇するかもしれないエラーについて説明します。

コードをコンパイル、リンクすると、CodeWarrior がエラーを発見することがあります。このときコンパイラまたはリンカがメッセージウィンドウにエラーを表示します。このマニュアルでは各エラーの意味、またそのエラーの解決方法を説明します。

エラーメッセージはその第 1 文字目の順に記載しています。シンボル名（変数や関数の名前など）、アルファベット以外の文字（句読記号など）、アルファベットの順番でエラーメッセージを説明します。

このマニュアルの各章はコンパイラ、リンカのエラーごとにまとめられています。

[『C/C++ コンパイラのエラーメッセージ』\(p7\)](#)

このマニュアルにおける表記方法

以下に、このマニュアルで使用するフォント、およびエラーの表記方法を説明します。

Error Message *variable name*

ここではエラーの性質とその原因を示します。エラーの発生時にコンパイラが発見した未知の変数名や型は斜体で示します。

)' expected

右括弧) が、正しい位置にありません。

注意： エラー、警告メッセージの例です。

リスト 1.1 ソースコードの例

```
In some cases, sample source code is provided that demonstrates the  
error message.
```

解決方法 エラーメッセージにその解決方法が含まれることもあります。

参考文献 詳細な情報を得るための参考文献を示します。ここでは主にエラーを検出、解消するための Metrowerks CodeWarrior の機能を紹介します。

エラーに影響する環境設定

言語オプションは、プログラムのソースコードのコンパイルだけでなく、エラーとみなすソースコードやコンパイルするソースコードにも影響します。多くの場合、エラーの出現は言語設定に原因があります。エラーの説明には、関連するチェックボックスの名前が含まれることがあります。

注意： エラーに影響するパネルについては『CodeWarrior IDE User Guide』を参照してください。

コンパイラの言語オプションを変更するには、Edit メニューの『Target Settings』を選択します。プロジェクト設定ダイアログが表れたら、左側のリストから適切な Language Settings 設定パネルを選択してください。



第 2 章 C/C++ コンパイラのエラー

PowerPC / 68K Macintosh、Win32/x86、Windows NT 用の Metrowerks CodeWarrior コンパイラの使用中に遭遇するかもしれないリンカエラーについて説明します。

C/C++ コンパイラのエラーメッセージ

エラーメッセージはその第 1 文字目の順に記載しています。シンボル名（変数や関数の名前など）、アルファベット以外の文字（句読記号など）、アルファベットの順番でエラーメッセージを説明します。

注意： C++ に関するエラーでは、「Margaret A Ellis and Bjarne Stroustrup. The Annotated C++ Reference Manual, Addison-Wesley, Reading, MA, 1990.」を参考として記載しているものがあります。この文献を ARM と略記し、参照ページを「ARM P202, 10.1.1 Ambiguities」という形式で掲載しています。

シンボル名 (C/C++)

シンボル名（変数、関数の名前など）で始まる C/C++ コンパイラのエラーメッセージを説明します。

エラー 10177 *class is not a SOM class*

SOM 以外のクラスにおいて SOM のプラグマを使っています。SOMReleaseOrder、SOMClassVersion、SOMMetaClass、SOMCallStyle などのプラグマは SOM クラスの中でのみ宣言可能です。

解決方法 SOM クラスの宣言の中にこれらのプラグマが使われていることを確認してください。これらのプラグマを SOMObject を継承していないクラスに使うことはできません。

エラー 10164 *variable could not be assigned to a register*

現時点では文書にされていません。

エラー 10051 *variable is not a struct/union/class member*

あるアイテムが構造体、共有体、クラスのメンバまたはメソッドとして参照されていますが、定義がありません。例えば、[リスト 2.1](#) では theColors.color はメンバとして参照されていますが、color は struct ColorValues 構造体のメンバではありません。

リスト 2.1 構造体、共有体、クラスのメンバではない

```
typedef struct
{
    short  seq; // color はこの構造体で
    short  group; // 定義されていない
} ColorValues;

ColorValues theColors;
theColors.color = 1; // error: 上記を参照
```

解決方法 問題のあった構造体、共有体、クラスを調べてください。このエラーは単純なスペルミス
が原因であることがあります。単純なスペルミスでないときは、アイテムの名前を変更す
るか、構造体を変更してください。

エラー 10193 *classname* is not an Objective-C class

classname が Objective-C クラスではありません。

句読記号 (C/C++)

句読記号で始まる C/C++ コンパイラのエラーメッセージについて説明します。

エラー 10126 *func* hides inherited virtual function *func2*

スーパークラスの仮想関数を隠すような非仮想メンバ関数を宣言しています。同じ名前で
異なる引数型をとる関数を宣言する場合、スーパークラスの関数は隠されてしまいます。例
を以下に示します。

リスト 2.2 継承した仮想関数 *func2* を *func* が隠している

```
class A {
public:
    virtual void f(int);
    virtual void g(int);
};

class B: public A {
public:
    void f(char); // WARNING: A::f(int) を隠す
    virtual void g(int); // OK: A::g(int) をオーバーライド
};
```

この警告は C/C++ Warnings 設定パネルの『Hidden virtual functions』オプションをオンにし
ているときのみ、表示されます。

解決方法 『Hidden virtual functions』 オプションをオフにするか、別の関数名を使ってください。

また、派生した仮想関数が基底仮想関数と同じ引数リストを使っているか確認してください。

リスト 2.3 関数宣言が継承した仮想関数を隠す

```
class X      { virtual void f(); };  
class Y : X { void f(int); };      // Y::f() が X::f() を隠す
```

エラー 10116 #if nesting overflow

#if のネストが許容範囲を越えています。

解決方法 #if を検証し、#if の大きなネストを小さなネストに変更してください。

エラー 10144 #include nesting overflow

#includes のネストが許容範囲を越えています。

解決方法 #include を検証し、#include の大きなネストを小さなネストに変更してください。

エラー 10015 '(' expected

コンパイラが予想しているところに (がありません。

解決方法 左右の括弧を確認するには『Balance』コマンドを使ってください。

エラー 10016 ')' expected

コンパイラが予想しているところに) がありません。

解決方法 左右の括弧を確認するには『Balance』コマンドを使ってください。前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。

このエラーを回避するために Editor Settings 環境設定パネルで『Balance While Typing』をオンにしてください。対応する左括弧がない右括弧を入力したとき、ピープ音が鳴ります。

参考文献 『Balance While Typing』については『CodeWarrior IDE User Guide』をご覧ください。

注意： シンタックスエラーや、直前のステートメントでのシンボルの消失がこのエラーの原因になることがあります。

エラー 10131 '<' expected

コンパイラが予想しているところに '<' がありません。

解決方法 前方のステートメントのシンタックスエラーまたはシンボルがないことが原因でこのエラーが発生する可能性があります。

注意：『Balance』コマンドは '<' または '>' では動作しません。

エラー 10132 '>' expected

コンパイラが予想しているところに '>' がありません。[リスト 2.4](#) に例を示します。

リスト 2.4 '>' が必要

```
template <class T> class Ca;  
CA *aClass;                                // error
```

解決方法 前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。

注意：『Balance While Typing』チェックボックスは '<' または '>' では動作しません。

参考文献 『Balance While Typing』については『CodeWarrior IDE User Guide』をご覧ください。

エラー 10017 ';' expected

コンパイラが予想しているところに ';' がありません。

解決方法 前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。例えば、[リスト 2.5](#) の `GetMenu()` でコンマがないというエラーが発生しますが、これは `GetMenu` 関数の後に ')' が欠けていることがエラーの原因です。

リスト 2.5 コンマがない

```
gAppleMenu = GetMenu(APPLE_MENU_ID);  
gFileMenu  = GetMenu(FILE_MENU_ID);  
// FILE_MENU_ID の後に ) がない
```

エラー 10071 ':' expected

コンパイラが予想しているところに ':' がありません。例えば、[リスト 2.6](#) の `switch` ステートメントでは `APPLE_MENU_ID` の後にコロンがありません。

リスト 2.6 コロンが必要

```
switch(theMenu)
{
    case APPLE_MENU_ID// error:  ':' がない

switch(theItem)
{
    case ABOUT_ITEM : // Correct
```

解決方法 前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。前方にあるエラーを修正してリコンパイルしてください。

エラー 10024 ';' expected

コンパイラが予想しているところに ';' がありません。例えば、[リスト 2.7](#) では WindowInit() の後にセミコロンがありません。

リスト 2.7 セミコロンが必要

```
ToolBoxInit();
WindowInit();// この行に ';' がない
MenuBarInit();
```

解決方法 前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。前方にあるエラーを修正してリコンパイルしてください。

エラー 10025 '[' expected

コンパイラが予想しているところに '[' がありません。

解決方法 左右の鍵括弧を確認するには、『Balance』コマンドを使ってください。前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。

このエラーを回避するために Editor Settings 環境設定パネルで『Balance While Typing』をオンにしてください。対応する左鍵括弧がない右鍵括弧を入力したとき、ピープ音が鳴ります。

参考文献 『Balance While Typing』については『IDE User Guide』を参照してください。

エラー 10026 ']' expected

コンパイラが予想しているところに ']' がありません。

解決方法 前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。前方にあるエラーを修正して再コンパイルしてください。

このエラーを回避するために Editor Settings 環境設定パネルで『Balance While Typing』をオンにしてください。対応する左鍵括弧がない右鍵括弧を入力したとき、ピープ音が鳴ります。

参考文献 『Balance While Typing』については『CodeWarrior IDE User Guide』を参照してください。

エラー 10036 '{' expected

コンパイラが予想しているところに '{' がありません。

解決方法 左右の中括弧を確認するには、『Balance』コマンドを使ってください。前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。

エラー 10031 '}' expected

コンパイラが予想しているところに '}' がありません。

解決方法 左右の中括弧を確認するには、『Balance』コマンドを使ってください。前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。

このエラーを回避するために Editor Settings 環境設定パネルで『Balance While Typing』をオンにしてください。対応する左中括弧がない右中括弧を入力したとき、ピープ音が鳴ります。

エラー 10157 '&' reference member *var* is not initialized

参照メンバが初期化されていません。すべての参照タイプはコンストラクタ関数のスコープ内で評価されなければなりません。[リスト 2.8](#) に誤った例と正しい例を示します。

リスト 2.8 リファレンスメンバの初期化

```
class caClass {
    private:
        int x;
    public
        const int &ref;
        caClass() {} // <-- 初期化されない
};
// 正しく初期化された参照
class caClass {
    private:
        int x;
    public:
        const int &ref;
        caClass():ref(x) {} //<-- 参照が初期化される
};
```

A ~ C (C/C++)

A、B、C で始まるエラーメッセージについて説明します。

エラー 10089 ambiguous access to class/struct/union member

クラス、構造体、共有体のメンバへの参照があいまいです。

仮想基底クラスと他の基底クラスで同じパラメータを持って定義された関数を呼び出すとき、このメッセージが出る可能性があります。CodeWarrior C++ コンパイラはこの種の状況に非常に厳格です。この場合、どちらの関数を使うのかをコード上で明確に示してください。

参考文献 ARM p.202, 10.1.1 Ambiguities.

エラー 10220 ambiguous access to name found '*symbol*' and '*symbol*'

名前へのアクセスがあいまいです。[リスト 2.9](#) のように同じ名前が複数回使われているときに起こるエラーです。

リスト 2.9 名前へのあいまいなアクセス

```
namespace A {  
    int a;  
}  
long a;  
namespace B {  
    using namespace ::A;  
    int x = a; //<<< ERROR: この 'a' は何か?  
}
```

エラー 10100 ambiguous access to overloaded function

多重定義関数 (overloaded function) への参照があいまいです。[リスト 2.10](#) の funcA() はデフォルトの引数を持つオーバーロード関数です。この関数が main() で呼び出されたとき、コンパイラはどちらのメンバ関数を使うべきかを判別できません。

注意: このエラーは double / float または short / long の引数を持つ関数をオーバーロードしたとき出やすいエラーです。

参考文献 ARM p.307, 13 Overloading.

リスト 2.10 多重定義関数へのあいまいな参照

```
class aClass  
{
```

```
        int x;
    public:
        int funcA( ) { x = 2; return x; }
        int funcA(int y = 3) { x = y; return x; }
};

main()
{
    aClass obj;

    obj.funcA();
    return 0;
}
```

エラー 10206 ambiguous message selector used: *msg* also had: *msg*

使用すべきセレクトタがあいまいです。

エラー 10247 ambiguous use of partial specialization

複数の部分的な特殊コードがクラス型に一致します。

エラー 10217 assigning a non-int numeric value to an unprototyped function

(警告) この警告は `#pragma warn_largeargs` がオンのとき、またはコマンドラインコンパイラに `warn_largeargs` を渡したときに発生します。

『require prototypes』オプションがオフのとき、非整数値 (float や longlong) をプロトタイプではない関数に渡したときにコンパイラが警告が発生します。

エラー 10186 assignment is not supported for SOM classes

SOMObject を継承したクラスに代入オペレーションを使うことはできません。SOM クラスはコピーコンストラクタ関数をサポートしていないためです。SOM オブジェクトの詳細は『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10059 branch out of range

アセンブラ関数内で範囲外への分岐が定義されています ([リスト 2.11](#))。

リスト 2.11 範囲外への分岐

```
bra.s 10000
```

エラー 10062 call of non-function

関数ではないものを関数呼び出ししています。例えば[リスト 2.12](#)では、変数 `i` を関数のように扱いエラーしています。

リスト 2.12 関数ではないものを関数呼び出しする

```
main()
{

    int i;
    ...
    i(); // error: "i" は関数ではない
    ...
}
```

解決方法 関数ではないものを調べてください。このエラーは、変数と似たような関数名をタイプミスした場合に発生します。

エラー 10228 cannot allocate initialized objects in the scratchpad

(PlayStationのみ)スクラッチパッドの静的な初期化はサポートされていません。[リスト2.13](#)のように初期化用ルーチンを呼び出してください。

リスト 2.13 スクラッチパッドに初期化オブジェクトを割り当てられない

```
__declspec(scratchpad) int P=1234; // ERROR
__declspec(scratchpad) int P;      // OK
...
void InitScratchPad()
{
    P = 1234;
}
```

エラー 10114 cannot construct base class *aClass*

基底クラス *aClass* に ctor 初期設定子またはデフォルトのコンストラクタ関数のいずれもありません。

エラー 10115 cannot construct direct member *aClass*

直接メンバ *aClass* に ctor 初期設定子またはデフォルトのコンストラクタ関数のいずれもありません。

エラー 10145 cannot convert from *Type_A* to *Type_B*

正しい変換コンストラクタ関数がないのに型変換を行おうとしています。または互換性のない型で変換しています。[リスト 2.14](#) は long * を int * として扱いエラーします。

リスト 2.14 *Type_A* を *Type_B* へ変換できない

```
main()
{
```

```
int *ptr;
ptr = new long; // <-- Error: 誤った型
return 0;
}
```

エラー 10154 cannot delete pointer to const

const 変数へのポインタは削除できません。[リスト 2.15](#)は const 変数へのポインタを削除しようとしてエラーします。

リスト 2.15 const 変数へのポインタを削除する

```
main()
{
    const int y = 3;
    int const *ptr = &y;
    delete ptr; // <-- Error here

    return 0;
}
```

エラー 10155 cannot destroy const object

const オブジェクトは削除できません。

エラー 10134 cannot instantiate *obj*

定義がなされていないため、テンプレート *obj* のインスタンスを生成できません ([リスト 2.16](#))

リスト 2.16 インスタンス生成できない

```
template <class T> class aClass;
template class aClass<int>;           // error
```

エラー 10137 cannot pass const/volatile data object to non-const/volatile member function

const 宣言されていないメンバ関数へ、const 宣言されたデータオブジェクトを渡しています。[リスト 2.17](#)ではエラーが起こります。

リスト 2.17 const データオブジェクトを非 const メンバ関数へ渡す例

```
struct stType {
    void bar();           // non-const メンバ関数
    void cbar() const;    // const メンバ関数
};
...
```



```
stType f;  
const stType cf;  
  
f.bar();    // OK  
f.cbar();   // OK  
cf.bar();   // error  
cf.cbar();  // OK
```

エラー 10151 cannot throw class with ambiguous base class *cBase*

基底クラスがあいまいなためクラスをスローできません。

解決方法 仮想基底クラスを宣言するか、あいまいな部分を解決してください。

エラー 10073 case constant defined more than once

switchステートメントで既に利用されたcase定数を再度使用しています。[リスト2.18](#)の例は定数 ABOUT_ITEM が2回使われています。

解決方法 どちらかの定数ラベルを削除してください。

リスト 2.18 同じ case 定数が2回呼び出されている

```
switch(theItem)  
{  
    case ABOUT_ITEM :  
        Alert(ABOUT_ALRT, NIL);  
        break;  
    case ABOUT_ITEM :  
        Alert(ABOUT_ALRT, NIL);  
        break;  
}
```

エラー 10143 'catch' expected

現時点では文書にされていません。

エラー 10156 const member *aVar* is not initialized

正しく初期化されていない const メンバが見つかりました。

解決方法 const メンバはオブジェクトを生成するときに初期化してください。

エラー 10212 category *Cat* redefined

既に定義されているカテゴリー *Cat* を、再度定義しようとしています。

エラー 10213 category *Cat* is undefined

定義されていないカテゴリー <Cat> を、使用しようとしています。

エラー 10196 class *classname* redeclared

既に定義されているクラス *classname* を、再度宣言しようとしています。

エラー 10197 class *classname* redefined

既に定義されているクラス *classname* を、再度定義しようとしています。

エラー 10104 class has no default constructor

デフォルトのコンストラクタ関数を持たないクラスを生成しようとしています ([リスト 2.19](#))。

リスト 2.19 デフォルトのコンストラクタ関数を持たないクラス

```
struct stType {  
    stType(int);  
};  
  
stType f; // error: デフォルトのコンストラクタ関数がない
```

エラー 10147 class type expected

クラスの型が必要です。

エラー 10125 'const' or '&' variable needs initializer

`const` または参照変数を宣言時に初期化しなければなりません。

リスト 2.20 `const` または参照変数の初期化が必要

```
const int a = 1; // OK  
const int b;    // ERROR: const 変数に初期設定子が必要  
  
int c;  
int &rc = c;    // OK  
int &rd;        // ERROR: 参照変数に初期設定子が必要
```

エラー 10159 constness casted away

`const` 型へ `volatile` 型をキャストしようとしています。

D ~ F (C/C++)

D、E、F で始まるエラーメッセージについて説明します。

エラー 10230 data object *object* redefined

データオブジェクトが不正に再定義されています。

リスト 2.21 データオブジェクト *<object>* の再定義

```
int a = 1;
int a = 2; //<<< ERROR: 変数名が不正に使われている
```

エラー 10046 data type is incomplete

クラスまたは構造体などのデータ型が不完全です。このエラーは前方宣言機能を使って宣言が分割されているようなクラスを使うときにしやすいエラーです。

参考文献 [『illegal use of incomplete struct/union/class』\(p42\)](#)

エラー 10022 declaration syntax error

宣言を解析中にシンタックスエラーが見つかりました。

解決方法 宣言を調べてください。エラーが特定できないときは、前方に何か問題があります。前にあるエラーを修正して再コンパイルしてください。

エラー 10035 declarator expected

宣言があることを予想していましたが、別のものが見つかりました。

解決方法 宣言を調べてください。エラーが特定できないときは、前方に何か問題があります。前にあるエラーを修正して再コンパイルしてください。

エラー 10074 default label defined more than once

同じ `switch` ステートメントの中に2つ以上の `default:` ラベルがありました。[リスト 2.22](#) に例を挙げます。

リスト 2.22 複数の `default:`

```
switch(...)
{
    default;;
    default; // switch に default は1つだけ
}
```

解決方法 余分な `default:` ラベルを削除してください。

エラー 10128 derived function differs from virtual base function in return type only

派生関数の戻り値の型が仮想基底関数の型と違います ([リスト 2.23](#))。

リスト 2.23 派生関数の戻り値の型が仮想基底関数の型と違う

```
class aClass { virtual int f(); }  
class bar : aClass {  
    void f();          // error  
}
```

エラー 10040 division by 0

定数式を 0 で除算、0 で剰余算しています。

エラー 10014 end of line expected

C/C++ Language 設定パネルの『ANSI Keywords only』がオンの場合、`#endif` 疑似命令の後にテキストがあるステートメントはエラーになります。ANSI 標準では `#endif` 疑似命令の後にはコメントしか書けません。このエラーはいろいろな場合に発生します([リスト 2.24](#))。

リスト 2.24 宣言文にトークンが足りない

```
#define // error: 両方の行に  
#if      // トークンが足りない
```

解決方法 多くの場合、`#endif` 疑似命令の後にコメント以外のテキストが続いているために、このエラーが発生します。そのときは、C/C++ Language 設定パネルの『ANSI Keywords only』をオフにして再コンパイルしてください。

エラー 10166 exception specification list mismatch

関数宣言と関数定義の例外処理仕様が合いません ([リスト 2.25](#))。

リスト 2.25 例外処理仕様のミスマッチ

```
void f() throw(int);  
        // 例外処理仕様が一致しない  
void f() throw(long)  
{  
}
```

exception handling does not work with `try` direct destruction 1

C++ 例外処理を使うときに、C/C++ Language 設定パネルの『Enable C++ Exception』をオフにしている、または `#pragma direct_destruction` をオンにしているとこのエラーが起こります。

解決方法 C/C++ Language 設定パネルの『Enable C++ Exception』をオンにする、または `#pragma direct_destruction` をオフにしてください。

エラー 10153 exception handling option is disabled

C/C++ Language 設定パネルの『Enable C++ Exception』がオフのときに、EH (throw など) を使おうとしています。

エラー 10042 expression syntax error

表現式のシンタックスが不当です。

解決方法 エラーが特定できないときは、前方に別のエラーがある可能性があります。前方のエラーを修正して再コンパイルしてください。

エラー 10066 function already has a stackframe

アセンブラ関数の中に2つ以上の `fralloc` 疑似命令があります。

解決方法 他の `fralloc` 疑似命令を削除してください。

エラー 10149 function call *func* does not match

引数が合いません。正しいコンストラクタ関数がない状況でオブジェクトを初期化すると、このようになることがあります。

解決方法 デフォルトのコンストラクタ関数をクラスへ追加してください。さらに、前で定義されているクラスや構造体を調べ、足りないオブジェクトのリストを見つけてください。

リスト 2.26 関数 '_func' が一致しない

```
class A {
    public:
    A() {}
} //<-- セミコロンがない、オブジェクトリストがない

main()
{
    A a; // <-- error が出る
    return 0;
}
```

エラー 10063 function call does not match prototype

関数呼び出しの引数がプロトタイプ宣言と一致しません。[リスト 2.27](#) では `SetFoo` 関数のプロトタイプ宣言は1つの引数ですが、`SetFoo` 関数の呼び出しは2つの引数を渡しています。

解決方法 関数呼び出しの引数を関数のプロトタイプ宣言と一致させてください。関数を選択し Search メニューで『Find Definition』を選択すると、関数のプロトタイプ宣言へ飛びます。

リスト 2.27 関数呼び出しがプロトタイプ宣言と一致しない

```
long SetFoo(long foonum)
{
    ...
}
...
MyFoo = SetFoo(size, length);
// error: SetFoo 呼び出しに 2 つの変数パラメータがある
//          プロトタイプの引数は 1 つだけ
```

エラー 10141 function defined 7 inline 1 after being called

関数が既に呼び出された後に、その関数が `inline` 宣言されています ([リスト 2.28](#))。

解決方法 関数のプロトタイプを `inline` 宣言してください。

リスト 2.28 呼び出された後に、`inline` 宣言されている関数

```
int func( int x );

class cA {
    int i;
public:
    cA() { i = func( 3); }
};

inline int func( int x ) { return x + 1; }
```

エラー 10067 function has no initialized stackframe

スタックフレームを確保 (`fralloc` 関数を使用) していないアセンブラ関数があります。

解決方法 `fralloc` を使ってアセンブラプログラムを修正してください。

エラー 10079 function has no prototype

関数プロトタイプを前方で宣言せずに関数を定義しています。C/C++ Language 設定パネルで『Require function prototypes』をオンにしている場合にそのような状況を発見したとき、このエラーが発生します。

解決方法 関数プロトタイプを宣言するか、C/C++ Language 設定パネルで『Require function prototypes』をオフにしてください。

参考文献 『Require function prototypes』については『C Compilers Reference』を参照してください。

エラー 10069 function nesting too complex

関数間でネストされた {} ブロックが多すぎます。

解決方法 問題がある関数を調べてください。関数をいくつかに分けることを考えてみてください。

エラー 10184 functions cannot return SOM classes

関数は SOMObject から継承されたクラスを返すことができません。SOM クラスはコピーコンストラクタ関数をサポートしていないためです。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10185 functions cannot have SOM class arguments

関数は引数に SOMObject から継承されたクラスを渡すことはできません。SOM クラスはコピーコンストラクタ関数をサポートしていないためです。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10227 function result is a pointer/reference to an automatic variable

自動変数を範囲外で参照している場合、このエラーが起こります ([リスト 2.29](#))。

解決方法 範囲を広げるか、変数の持続性を高めてください。例えば、スタックの局所変数ではなく、ヒープのポインタとして変数を宣言してください。

リスト 2.29 自動変数へのポインタ / 参照

```
char *foo()
{
    char c=0;
    return &c;
}
```

G ~ I (C/C++)

G、H、I で始まるエラーメッセージについて説明します

エラー 10189 global SOM class objects are not supported

オブジェクトが SOMObject を継承する型である場合、そのオブジェクトを大域変数として使用できません。SOM クラスはコピーコンストラクタ関数をサポートしていないからです。SOM オブジェクトの詳細については「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10023 identifier *name* redeclared

現時点では文書にされていません。

エラー 10150 identifier *name* redeclared was declared as: *a_type* I now declared as: '*b_type*

識別子が二重に定義されています。[リスト 2.30](#) の例では識別子 `var` が `Point` と `long` の両方の型で宣言されています。

注意： これは Macintosh の Toolbox 関数の名前をクラス名として使用しようとする場合によく出るエラーです。

解決方法 識別子の名前を変更してください。

リスト 2.30 識別子 `var` の二重定義

```
long    var;
Double  temp;
Point   var;
        // var は既に宣言されている

// Macintosh Toolbox 関数の名前をクラス名として使用している

class Random { public: void get(); };

void Random::get() // <-- ここでエラー
{ //... }
```

エラー 10008 identifier expected

識別子のトークンが足りません。例えば、[リスト 2.31](#) では識別子があるべきところに `short` があるためにエラーとなります。

解決方法 エラーの内容を特定できないときは、前方の行でシンボル (`;` や `,` など) がないことが原因の可能性があります。前方にあるエラーを修正して再コンパイルしてください。

リスト 2.31 識別子がない

```
long Waggle;
longtemp, short; // 識別子ではなく short がある
PointmyPt;
```

エラー 10087 illegal #pragma

不当な `#pragma` 疑似命令です。

解決方法 Metrowerks C が認識できる `#pragma` 疑似命令のリストは、『C Compilers Reference』に記載されています。正しい `#pragma` 疑似命令を使っていることを確認してください。

エラー 10097 illegal '&' reference

不当なポインタ参照をしています ([リスト 2.32](#))。

リスト 2.32 不当な '&' 参照

```
int &&g; // error: integer を 2 回間接参照できない
```

エラー 10231 illegal access to local variable from other function

他の関数から局所変数へ不当なアクセスをしています。

リスト 2.33 局所変数への不当なアクセス

```
void bar()  
{  
    int a=0;  
  
    struct nest {  
        void foo()  
        {  
            a=2; //<<< ERROR: 'a' は bar の局所変数  
                //      nest::foo からはアクセスできない  
        }  
    };  
}
```

エラー 10142 illegal constructor/destructor declaration

コンストラクタ関数またはデストラクタの宣言方法が間違っています。オブジェクトが既に初期化された後、コンストラクタ関数を呼び出している例を[リスト 2.34](#)に示します。

リスト 2.34 コンストラクタ関数の不当な宣言

```
class cA {  
public:  
    void cA() {}  
};  
  
cA A;  
  
main() {  
    A.cA();  
    return 0;  
}
```

エラー 10160 illegal const/volatile '&' reference initialization

`const` または `volatile` の参照が正しく初期化されていません。

解決方法 オブジェクトの生成中に参照を初期化していないことが原因であることが多いです。

エラー 10082 illegal data in precompiled header

プリコンパイルヘッダに不当なデータが入っています。

解決方法 プリコンパイルヘッダからデータを取り除き、再度プリコンパイルしてください。プリコンパイルヘッダの不当なデータについては『C Compilers Reference』を参照してください。

エラー 10165 illegal exception specification

許可されていない例外処理仕様を使っています ([リスト 2.35](#))。

リスト 2.35 不当な例外処理仕様

```
typedef void (*f)() throw(int);  
// typedef で spec は使えない
```

エラー 10148 illegal explicit conversion from *type_A* to *type_B*

ある型を不当な型へ変換しようとしています。[リスト 2.36](#) では、クラスへのポインタを `long` 型へ変換しています。

リスト 2.36 不当な明示的変換

```
class D { };  
  
main()  
{  
    long x;  
    D d;  
    x = (long)d;  
    return 0;  
}
```

エラー 10102 illegal 'friend' declaration

不当な `friend` 宣言です。[リスト 2.37](#) では `friend` が不当に宣言されています。

リスト 2.37 不当な `friend` 宣言

```
int i;  
class aClass {
```

```
    friend i; // 不当な 'friend' 宣言  
};
```

エラー 10103 illegal 'inline' function definition

既に参照されている関数が `inline` 関数として定義されています。

エラー 10094 illegal 'operator' declaration

不当な `operator` 宣言です ([リスト 2.38](#))。

リスト 2.38 不当な 'operator' 宣言

```
int operator +(int,int,int);
```

エラー 10140 illegal `class::constructor(argument)` copy constructor

同じクラス型の引数を 1 つ持つようなクラスのコンストラクタ関数を定義してます ([リスト 2.39](#))。

リスト 2.39 不当な `class::constructor(argument)` のコピーコンストラクタ関数

```
class aClass {  
    aClass(aClass); // error  
    aClass(aClass&); // OK  
};
```

エラー 10101 illegal access/using declaration

不当なアクセス宣言です。このエラーは派生クラスから基底クラスへのアクセスが正しく宣言されていないことを示します。派生クラス宣言の公開または限定公開部で基底クラスの限定クラス名を使って、基底クラスへのアクセスを修正できます ([リスト 2.40](#))。

参考文献 ARM p. 244, 11.3 Access Declaration.

リスト 2.40 不当なアクセス宣言

```
class B {  
    private:  
    int a  
    public:  
    int b;  
};  
  
class D : private B {
```

```
    public:
    B::a; // <-- Error 不当なアクセス宣言
    B::b; // 外部関数での B::b の使用は可
    int x;
    void fx() { b = x; }
}Derived;

inline void fx() { Derived.b = 3; }
```

エラー 10093 illegal access qualifier

不当な指定子です。[リスト 2.41](#) では、foo:: をクラス指定子として定義していないため、エラーとなります。

参考文献 ARM p. 112.

リスト 2.41 不当なアクセス指定子

```
struct stType { enum efoo { nfoo } mfoo; };
...
foo::xfoo; // error: 不当な指定子
```

エラー 10088 illegal access to protected/private member

該当関数は限定公開 / 非公開メンバへアクセスできません。[リスト 2.42](#) では priv は private 宣言されています。func() 内で非公開メンバへアクセスしたエラーです。

リスト 2.42 非公開メンバへの不当なアクセス

```
class aClass { private: int priv; }
func() {
    aClass x;
    x.priv=0;
    //func() は非公開メンバ priv へアクセスできない
}
```

エラー 10056 illegal addressing mode

アセンブラ命令で使用不可能なアドレッシングモードです ([リスト 2.43](#))。

リスト 2.43 不当なアドレッシングモード

```
moveq d0,d1
```

エラー 10010 illegal argument list

不当なマクロの引数です ([リスト 2.44](#))。

リスト 2.44 不当な引数

```
#define macro(arg,,)
```

エラー 10030 illegal array definition

配列定義に負数や 0 を指定しています。[リスト 2.45](#) のような空の配列は ANSI 以外の C の拡張ですが、Metrowerks CodeWarrior C/C++ ではサポートされていません。エラーになります。

解決方法 [リスト 2.45](#) のような配列は、[リスト 2.46](#) のように修正してください。

リスト 2.45 不当な配列定義

```
typedef struct {  
    short howMany;  
    Data *dataBase[]; // error: ANSI 拡張ではない  
} DataBase
```

注意: [リスト 2.46](#) のような変更をしたときは、メモリを取得するルーチンも変更してください。例えば、リスト 2.40 の場合の正しいメモリ取得ルーチンを以下に示します。

```
sizeof(DataBase) - (nb_elements - 1) * sizeof(Data).
```

リスト 2.46 不当な配列定義を修正

```
typedef struct {  
    short howMany;  
    Data *dataBase[1]; // OK: ANSI 準拠である  
} DataBase
```

エラー 10080 illegal assignment to constant

const 型の変数に値を代入しています ([リスト 2.47](#))。

リスト 2.47 const 型の変数への代入

```
const int i=5;  
...  
i=10; // const に代入できない
```

解決方法 代入文を調べてください。入力すべき変数名が `const` 型の変数と似ているため、入力ミスをしているかもしれません。

エラー 10039 illegal bitfield declaration

ビットフィールドのサイズの宣言が 0 である、または大きすぎます。

解決方法 ビットフィールド宣言を調べ、そのサイズが 0 でないこと、大きすぎないことを確認してください。

リスト 2.48 不当なビットフィールド宣言

```
long err:33;
```

エラー 10001 illegal character constant

不当な文字定数です。[リスト 2.49](#) では不当な文字定数の例を 3 つあげています。

解決方法 文字定数が文法的に正しいかどうかを確認してください。

リスト 2.49 不当な文字定数

```
char FooCh;  
...  
...  
  
FooCh = ' '; // これらの代入には  
FooCh = '\x'; // 不当なキャラクタ定数が  
FooCh = 'ddjddjdj'; // 含まれている
```

エラー 10229 illegal class member access

存在しないクラスメンバへアクセスしています。

エラー 10025 illegal constant expression

定数式に不当な値またはオペレータがあります。

解決方法 定数式を調べて修正してください。このエラーが特定できないときは、前方にエラーがあるかもしれません。前方にあるエラーを修正して再コンパイルしてください。

エラー 10113 illegal ctor initializer

不当な `ctor` 初期設定子です。例えば、[リスト 2.50](#) の `ctor` 初期設定子は `i` メンバがないため、正しくありません。

解決方法 サブクラスのコンストラクタ関数で何か問題がある場合は、親クラスの名前を明示的に付けていないことが原因として考えられます。[リスト 2.51](#) に例を示します。

リスト 2.50 不当な ctor 設定子

```
class aClass {  
    aClass(): i(12) {} // error: 不当な ctor  
                        // (i メンバがない)  
};
```

CButtonはDisplaySystem*を引数にとるClickableのサブクラスです。CodeWarriorのC++仕様では、親クラスに引数を渡すコンストラクタを作成するとき、親クラスの名前を明示的に示す必要があります。以下のテンプレートを使用しなければなりません。

```
BLAH::BLAH(parameter1, parameter2) :  
    PARENT_OF_BLAH(parameter2)
```

リスト 2.51 明示的に親クラスの名前を定義していない

```
Clickable::CButton(long resourceBase,  
    DisplaySystem* displaySystem) : (displaySystem)
```

エラー 10057 illegal data size

アセンブラ関数内に不当なデータサイズがあります ([リスト 2.52](#))。

リスト 2.52 不当なデータサイズ

```
move.Z #0,d9
```

エラー 10106 illegal default argument(s)

不当な引数を含んでいる関数呼び出しがあります ([リスト 2.53](#))。

リスト 2.53 不当なデフォルト引数

```
int func(int x=1, int z);
```

エラー 10117 illegal empty declaration

宣言に識別子がありません ([リスト 2.54](#))。

リスト 2.54 空の宣言

```
int ;
```

エラー 10139 illegal explicit template instantiation

不当な明示的テンプレートインスタンス生成です。[リスト 2.55](#)では、関数 `f()` が、テンプレート関数として正確に宣言されていません。

リスト 2.55 不当な明示的テンプレートインスタンス生成

```
//template <class T>
void f();
template void f<int>();      // error
```

エラー 10236 illegal explicit template specification

明示的テンプレート仕様に誤りがあります。

```
template <> int a;
```

エラー 10028 illegal function definition

不当な関数定義です。

解決方法 このエラーが特定できないときは、前方にエラーがあるかもしれません。前方にあるエラーを修正して再コンパイルしてください。

エラー 10098 illegal function overloading

関数が不当に多重定義されています。通常、このエラーは同じ名前と同じ引数を持ち、型を返すような関数を宣言したときに発生します。

参考文献 ARM p.307, 13 Overloading.

エラー 10029 illegal function return type

配列および関数を返すような関数があります。関数は配列や関数を返すことはできません。関数は配列へのポインタおよび関数へのポインタを返すことはできます。

解決方法 該当する関数を関数へのポインタまたは配列へのポインタを返すように修正してください。

エラー 10121 illegal implicit const pointer conversion

(警告) `const` ポインタを変数へ変換しています。

リスト 2.56 不当な暗示的 `const` ポインタ変換

```
void func(const char *cptr)
{
```



```
char *ptr=cptr; // 不当な暗示的 const ポインタ変換  
}
```

エラー 10110 illegal implicit conversion from *Type_A* to *Type_B*

[リスト 2.57](#) のような不当な暗示的（キャストしない）型変換をしています。

注意： ANSI C++ と ANSI C では void* の扱い方が違います。ANSI C はポインタから void 型への暗示的な型変換を許します（ただし、関数へのポインタについては許容されません）。C++ では、void* への明示的なキャストなしでは void* 以外のオブジェクトへ適用できません。[リスト 2.57](#) に ANSI C では許容されても、C++ ではエラーとなる構文の例を示します。

参考文献 ANSI Draft Standard Section 5.4 “Pointer Conversions,” ANSI C Standard 3.2.2.3

リスト 2.57 不当な暗示的型変換

```
void f(char *cptr, void *vptr)  
{  
    cptr = vptr;  
    //C++ では不当、C では正当  
    char *ptr=(void *)0;  
    //C では不当、C++ では正当  
    // . . .  
}
```

エラー 10118 illegal implicit enum conversion from *Type_A* to *Type_B*

enum 型への暗示的な型変換をしています。[リスト 2.58](#) では整数を enum 型へ変換するというエラーがあります。ソースが C++ である場合、コンパイラはこのメッセージをエラーとして表示します。ソースが C で、かつ C/C++ Warnings 設定パネルの『Extended Error Checking』がオンの場合、コンパイラはこのメッセージを警告として表示します。

リスト 2.58 不当な暗示的 enum 型変換

```
enum ff { foo };  
enum ff x = 0;  
//error: 不当な暗示的 enum 型変換
```

エラー 10232 illegal implicit member pointer conversion

メンバ関数が正しく初期化されていません。

リスト 2.59 不当な暗示的メンバポインタの変換

```
struct X {  
    void foo()  
    {  
        void (X::*f)() = foo;    // ERROR  
        void (X::*g)() = &X::foo; // OK  
    }  
};
```

エラー 10075 illegal initialization

変数および他のデータ型に対して、関数内で不当な初期化がされています。

エラー 10053 illegal instruction for this processor

アセンブラ命令の中に 68K ファミリのマイクロプロセッサにはない命令が指定されています。

エラー 10112 illegal jump past initializer

ブロック内にある初期化ルーチンを実行せずに外からブロック内へ分岐しています。このエラーが発生する典型的な個所は、switch ステートメントまたは goto ステートメントです ([リスト 2.60](#))。

注意： 変数を switch ステートメントの外で定義するか、新しいスコープを作成してください。関数の中に goto ステートメントがあるとき、goto ステートメントの後ろで定義された変数でこのエラーが発生します。対策は、switch ステートメントと同様、すべての変数を goto ステートメントの前で宣言するか、新しいスコープを作成してください。

参考文献 ARM p. 87, 6.4.2 The Switch Statement and p.91, 6.7 Declaration Statement.

リスト 2.60 不当な jump past の初期化

```
switch (i) {  
    int v1 = 2; // error  
    case 1:  
        short v2 = 3;  
    case 2:  
        if( v2 == 7 ) {} // error  
}
```

エラー 10204 illegal message receiver

Objective-C オブジェクト以外のオブジェクトにメッセージを送信しようとしています。

エラー 10221 illegal namespace

ネームスペースでそれ以外の名前を使用しています。

リスト 2.61 不当なネームスペース

```
int a;  
namespace a { //<<< ERROR  
    int b;  
}
```

エラー 10223 illegal name overloading

不当なオーバーロードが行われています。

リスト 2.62 名前の不当なオーバーロード

```
int b;  
enum { b } //<<< ERROR
```

エラー 10045 illegal operand

オペレータが互換性のないオペランドを見つけました。

解決方法 互換性のある型にするためにオペランドをキャストしてください。

エラー 10054 illegal operands for this processor

68K ファミリのマイクロプロセッサにないオペランドを参照する命令です。

エラー 10044 illegal operation

==、+ のようなオペレータが不当に構造体や共有体へ適用されました。または、オペレータにデータ型が定義されていません。

エラー 10105 illegal operator

不当なオペレータです。

リスト 2.63 不当なオペレータ

```
int operator .(int i);
```

エラー 10099 illegal operator overloading

オペレータが不当に多重定義されています。このエラーは、多重定義できない演算子を多重定義しようとしたり、プリプロセッサ命令を多重定義しようとしたときに発生します。

参考文献 ARM p.329, 13.4 Overloaded Operators.

エラー 10245 illegal partial specialization

部分的なクラスの特異化コードにエラーがあります。

エラー 10246 illegal partial specialization argument list

部分的なクラスの特異化コードの引数リストにエラーがあります。

エラー 10124 illegal precompiled header compiler flags or target

プリコンパイルヘッダが間違ったターゲットを使っています。例えば、PowerMac Target のプリコンパイルヘッダの 1 行目に # include <MacHeaders68K> と書かれているような場合です。

解決方法 プリコンパイルヘッダ、または .pch ファイルのフラグや CPU タイプとカレントプロジェクトのそれが適合するかどうかを調べてください。C/C++ Language 設定パネルの『Prefix File』フィールドの名前も調べてください

エラー 10123 illegal precompiled header version

プリコンパイルヘッダが古いか、何かが足りません。

解決方法 以下の方法を試してください。

プロジェクトのプリコンパイルヘッダを再度プリコンパイルしてください。

Metrowerks のプリフィックスファイルを使用している場合、プリコンパイルヘッダプロジェクトをリビルドしてください (MacHeaders.mcp、Win32Headers.mcp、MSLHeaders.mcp を含むもの)。

エラー 10058 illegal register list

アセンブラ関数の中に不当なレジスタリストがあります。[リスト 2.64](#) では d0-d0 という正しくないレジスタリストを指定しています。

リスト 2.64 不当なレジスタリスト

```
movem.l d0-d0, -(sp)
```

エラー 10216 illegal return value in void/constructor/destructor function

void 関数、コンストラクタ関数、デストラクタ関数のいずれかから値を戻そうとしています。これらは値を戻すことができません。

解決方法 不当な戻り値を除去してください。

リスト 2.65 不当な戻り値

```
void foo() { return 1; }
```

エラー 10171 illegal SOM function overload *operator*

SOM クラスのメンバ関数はオーバーロードできません。SOM オブジェクトの詳細は『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10174 illegal SOM function parameters or return type

SOM オブジェクトに long double の引数または戻り値は使えません。SOM オブジェクトの詳細は『C Compilers Reference』 または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 異なる引数型または戻り値の型を使って関数を定義しなおしてください。

エラー 10078 illegal storage class

不当なストレージクラスです。

ARM, p179 には、『クラスのデータおよびメンバ関数は、クラス宣言の中で static 宣言されなければならない』と書かれています。同様に、グローバルスコープにおいて auto 変数を宣言するとエラーになります ([リスト 2.66](#))。

参考文献 ARM , p.179.

リスト 2.66 不当なストレージクラス

```
auto int x;//error: グローバルスコープで auto は不可
```

エラー 10002 illegal string constant

行中に完結していない文字列定数があります。

解決方法 改行する前に文字列定数を完結させてください。このエラーを特定できないとき、前方に別のエラーがあるかもしれません。前方のエラーを修正してリコンパイルしてください。

エラー 10032 illegal struct/union/enum/class definition

不当な構造体、共有体、列挙型、クラスの定義です。

解決方法 シンタックスエラーを調べてください。

エラー 10133 illegal template argument(s)

テンプレートに不当な引数があります。

解決方法 このエラーは> (右シフト) オペレータの書き忘れが原因でしばしば発生します。

リスト 2.67 テンプレートの不当な引数

```
map<double, double, less<double>> aMap;  
    // 不当な引数  
map<double, double, less<double> > aMap;  
  
template <class T> class aClass;  
aClass<int,int> *aClassptr; // 不当な引数
```

エラー 10130 illegal template declaration

不当なテンプレート宣言のフォーマットです。

リスト 2.68 不当なテンプレート宣言

```
template <class T> T i;           // error
```

エラー 10006 illegal token

不当なプリプロセッサトークンがあります。[リスト 2.69](#) では、@ が不当なトークンです。

解決方法 不当なトークンを削除してください。このエラーを特定できないとき、前方に別のエラーがあるかもしれません。前方のエラーを修正して再コンパイルしてください。

リスト 2.69 不当な token (@シンボル)

```
if( @ )
```

エラー 10047 illegal type

[リスト 2.70](#) のような不当なタイプがあります。

リスト 2.70 不当なタイプ

```
static void func(int i)
{
    delete i; // <-- 不当なタイプ
}
```

エラー 10065 illegal type cast

表現式が互換性のない型でキャストされました。

エラー 10077 illegal type qualifier(s)

範囲内の型に不当な型指定子があります。[リスト 2.71](#) では `const` 指定子が 2 つあるため、エラーが起こります。

解決方法 不当な型指定子を削除してください。このエラーを特定できないとき、前方に別のエラーがあるかもしれません。前方のエラーを修正して再コンパイルしてください。

リスト 2.71 不当な型指定子

```
const const int x; // const が重複
```

エラー 10214 illegal use of *spec*

不当なシンタックスのキーワードが使われています。主に指定子、修飾子で発生するエラーです。

リスト 2.72 不当な *spec* の使用

```
inline int k;           // 'inline' の不当な使用
const int const cc;     // 'const' の不当な使用
```

エラー 10242 illegal use of `alloca()` in function argument

(68K コンパイラのみ) 68K バージョンの `alloca()` を関数引数の表現式で使うことはできません。

```
#include <alloca.h>

extern void foo(void*,int);

static void bar()
{
    void *ptr;
```

```
        foo(alloca(100),1);    // error

        ptr=alloca(100);
        foo(ptr,1);            // OK
    }
```

エラー 10239 illegal use of asm inline function

エントリ、または PC 相対データをインラインアセンブラ関数で使用的是います。

リスト 2.73 インラインアセンブラ関数の不当な使用例

```
extern void e();
inline asm int GetD7()
{
    move.l d7,d0
    entry e
}
```

エラー 10240 illegal use of C++ feature in EC++

EC++ 言語サブセットで、使用不可能な C++ 機能（テンプレート、ネームスペース、多重継承など）を使用しています。

エラー 10222 illegal use of namespace name

ネームスペースをそれ以外の用途に使用しています。

リスト 2.74 ネームスペースの不要な使用

```
namespace a {
    int b;
}
int g = a++; //<<< ERROR
```

エラー 10208 illegal use of Objective-C object

ObjC クラスを、サポートされていない方法で使用的是います。例えば値でクラスを渡す、ObjC クラスオブジェクトを宣言 / 定義する、などです。

エラー 10178 illegal use of #pragma outside of SOM class definition

SOM プラグマ (SOMReleaseOrder、SOMClassVersion、SOMMetaClass、SOMCallStyle) は SOM クラスの定義内でのみ使用可能です。SOM オブジェクトの詳細は『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10119 illegal use of #pragma parameter

関数の前にある #pragma 宣言の引数が関数名と一致しません。例えば、[リスト 2.75](#) の func() では #pragma の引数として func1 ではなく間違えて func を指定しています。

注意： C/C++ Warnings 設定パネルで『Illegal Pragmas』がオンになっているとき、未定義の #pragma は警告となります。

参考文献 Metrowerks C/C++ でサポートされる #pragma については『C Compilers Reference』の第6章、また C/C++ Warnings 設定パネルの詳細は第2章を参照してください

リスト 2.75 不当な #pragma の引数

```
#pragma parameter __A0 func
char *func1()
{
    // ...
}
```

エラー 10092 illegal use of 'HandleObject'

HandleObject から派生したサブクラスを不当に使用しています。

エラー 10202 illegal use of 'self'

キーワード self を間違えて使用しています。

エラー 10203 illegal use of 'super'

キーワード super を間違えて使用しています。

エラー 10090 illegal use of 'this'

C++ コードで、非メンバ関数で this を使用しています。

エラー 10027 illegal use of 'void'

オペレータが誤って void 型になっている、または変数が void 宣言されています ([リスト 2.76](#))

リスト 2.76 不当な void 使用例

```
int myInt;
long myNumber;
void myFoo;
//error: 変数は void 宣言できない
```

エラー 10095 illegal use of abstract class (*aClass*)

抽象クラスが不当に使われています。抽象クラスは、最低 1 つの純粋仮想メソッドを定義しなければなりません。純粋仮想メソッドは、以下のように定義されます。

```
// pure virtual method
virtual type MethodName ( arguments ) = 0;
```

抽象クラスには、純粋仮想メソッドを書き換えるサブクラスが必要です。

解決方法 抽象クラスは、実体化する前にサブクラスを定義しなければなりません。抽象クラスではないサブクラスを定義し、そのサブクラスから自分のオブジェクトを派生してください。

参考文献 ARM p. 214, 10.3 Abstract Classes.

エラー 10084 illegal use of direct parameters

サポートされていない直接引数を使用しています。__D0 から __D2 まで、__A0、__A1、および、__FP0 から __FP3 までが直接引数としてサポートされています。

エラー 10037 illegal use of incomplete struct/union/class

不完全な構造体、共有体、クラス定義が使われています。[リスト 2.77](#) の例は、`struct A` の定義が不完全なのにオブジェクトを宣言しようとしてエラーが発生します。

解決方法 エラーを回避するためには、`bar.h` をインクルードします（クラスの宣言を完全に行うために必要です）。インクルードファイルに循環参照があるとき、上記の例のような参照の仕方はエラーの原因となります。このエラーの別の回避方法に、部分クラスのメンバ変数やメソッドへのインライン参照を行わない（ファイルをインクルードしない）という方法もあります。これらは `foo.h` と `bar.h` の両方を含む実装ファイルに入れてください。

リスト 2.77 不完全な構造体を不当に使用

```
struct A x; // 不完全なオブジェクトを作成できない
```

不完全なクラスによるエラーは（通常は以下の例のように前方参照を使用した）部分的に宣言されたクラスを使用することによって発生します。

リスト 2.78 不完全なクラスを使用

```
// foo.h
class Bar; // 空の forward 宣言

class aClass {
public:
    // トラブル
    void DoIt(int x) { mBar->DoStuff(x); }
    ...
private:
```

```
    Bar* mBar; // 宣言
};

bar.hclass Bar { // クラスの実際の宣言
public:
    void DoStuff(int x);
    ...
};
```

エラー 10076 illegal use of inline function

インライン関数の使い方が不当です。例えば、[リスト 2.79](#) はインライン関数のアドレスを求めています。

リスト 2.79 インライン関数の不当な使用

```
pascal Handle NewHandle(Size byteCount) = 0xA122;
...
&NewHandle; // error: インライン関数のアドレスは取れない
```

エラー 10070 illegal use of keyword

キーワードの使い方が不当です。前方のステートメントのシンタックスエラーが原因でこのエラーが発生する可能性があります。例えば、[リスト 2.80](#) では ; がいないためエラーが発生します。

解決方法 前方にあるエラーを修正してください。それでもエラーが出るときは、[リスト 2.81](#) のように、キーワードが正しいかどうかを調べてください。

リスト 2.80 キーワードの不当な使用

```
switch(theMenu)
{
    case APPLE_MENU_ID // error: ':' がない
    {
        switch(theItem)
        {
            case ABOUT_ITEM :
                ...
                ...
        }
        break; // 上でコロンがないため、
               // キーワードの不当な使用のエラー
    }
}
```

リスト 2.81 直接パラメータの不当な使用

```
int func(int x: __X) {}  
// error: 直接パラメータ __X が認識できない
```

エラー 10122 illegal use of non-static member

クラスのオブジェクト名を指定しないで非静的メンバ関数へアクセスしています。

リスト 2.82 非静的メンバ関数の不当な使用

```
struct stType {  
    stTypef();  
};  
// ...  
    stTypef();  
// error: stType オブジェクトなしでは呼び出しできない
```

エラー 10081 illegal use of precompiled header

プリコンパイルヘッダが不当にインクルードされました。[リスト 2.83](#) では、プリコンパイルヘッダが複数回インクルードされています。

解決方法 プロジェクトに含まれているすべてのプリコンパイルヘッダファイルを調べてください。エラーを特定できないときは、C/C++ Languages 環境設定の『Prefix File』フィールドを調べてください。

リスト 2.83 同じプリコンパイルヘッダをインクルードするエラー

```
#include <MacHeaders>  
#include <MacHeaders> // error: プリコンパイルヘッダは 1 つだけ許可される
```

解決方法 C/C++ Language 環境設定の『Prefix file』フィールドで指定されているプリコンパイルヘッダをインクルードしています。関数、変数、型宣言の後でプリコンパイルヘッダをインクルードしています ([リスト 2.84](#))。

リスト 2.84 変数宣言の後でプリコンパイルヘッダをインクルードするエラー

```
#include <MacHeaders>  
// error: プリコンパイルヘッダが以下の宣言で  
//          インクルードされている
```

エラー 10096 illegal use of pure function

純粋仮想関数の使い方が不当です。[リスト 2.85](#) では、コンストラクタ関数が純粋関数 `myFun()` を呼び出そうとしています。

参考文献 ARM, p. 214, 10.3, Abstract Classes.

リスト 2.85 純粋関数の不当な使い方

```
class pure {
public:
    virtual int myFun() = 0;
    pure() { myFun(); }
};
```

エラー 10064 illegal use of register variable

レジスタ変数の使い方が不当です。[リスト 2.86](#) はレジスタ変数のアドレスを指定するというエラーです。

リスト 2.86 レジスタ変数の不当な使用

```
register int i;
f(&i);    // error: i のアドレスが取れない
```

エラー 10241 illegal use template argument dependent type 'T::%u'

テンプレートの依存型が解析できません ([リスト 2.87](#))。

リスト 2.87 テンプレートの引数依存型 'T::%u' の不当な使用

```
template <class T> int foo(typename T::x arg);
int i = foo<int>(1);
// error: テンプレートの引数依存型 'T::x' の不当な使用
```

エラー 10218 implicit arithmetic conversion from *Type_A* to *Type_B*

大きな算術型を小さいものへ暗示的に変換するときこの警告が表れます (Warnings 設定パネルで設定します)。

```
long l;
short s;
...
s=l; // <<<error
```

エラー 10250 implicit 'int' is no longer supported in C++

C++ コードで暗黙の `int` を使用したときにこのエラーが発生します。これは C++ ではサポートされていません。

```
main() // <<< int main() へ変換しなくてはならない
```

エラー 10161 inconsistent linkage: 'extern' object redeclared as 'static'

C++ で `extern` オブジェクトを `static` で再宣言したり、再定義したとき、このエラーが発生します ([リスト 2.88](#))。

リスト 2.88 非整合 : "extern" オブジェクトを "static" 宣言する

```
void f();  
static void f(); // <<< ERROR
```

エラー 10244 inconsistent use of 'class' and 'struct' keywords

`class` と `struct` キーワードの使い方が一致していません。

以下の例では、`X` は最初に `class` 宣言されていますが、次に `struct` 宣言されています。

```
#pragma warn_structclass on  
class X;  
struct X { int a; }; // 警告 : 'class' と 'struct' キーワードの  
// 使用の不一致
```

エラー 10243 inline function call *function name* not inlined

関数がインラインできないとき (インラインレベルが低いときなど) このエラーが発生します。

エラー 10224 instance variable list does not match @interface

インターフェースで宣言されたインスタンス変数リストが、実装と一致していません。

リスト 2.89 @interface とインスタンス変数が一致しない

```
@interface A  
{  
    int a;  
}  
@end  
@implementation A  
{
```

```
    long b; //<<< ERROR: 宣言と一致しない
}
@end
```

エラー 10179 introduced method *method* is not specified in release order list

SOM クラスに `SOMReleaseOrder` のプラグマ命令文を使うとき、このプラグマ命令文はクラスが宣言している（オーバーライドするメソッドではない）すべての `new` メソッドをリストしなければなりません。SOM オブジェクトの詳細は『C Compilers Reference』または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 この問題を解決するには2つの方法があります。

`SOMReleaseOrder` のリストにメソッドを含めてください。

`SOMReleaseOrder` を削除してください。コンパイラはリリース順はクラス宣言で関数が現われた順番と同じであると仮定しています。しかし、クラスのバージョンをリリースするとき、プラグマを使います。最新バージョンのクラスにおいてリストの内容を変更したい場合があるからです。

J ~ L (C/C++)

J、K、L で始まるエラーメッセージについて説明します。

エラー 10072 label *Lgt* redefined

既に定義されたラベルを再定義しています（このメッセージはラベル *Lgt* が既に定義されているという意味です）。

解決方法 該当ラベルを削除するか、名前を変更してください。

エラー 10248 local classes shall not have member templates

ローカルクラスにメンバテンプレートがあります。

```
void f()
{
    struct X {
        template <class X> void f(X) {}; // error
    }
}
```

エラー 10111 local data >32k

ローカルデータの総数が 32K バイトを越えました。ローカルデータ（通常宣言された配列）は 32K の制限があるスタックに保存されます。

解決方法 配列を静的変数として確保するか、(スタックではない) ヒープ上にダイナミックにメモリ領域を確保してください。

エラー 10163 local data > 224 bytes

(Mac OS PowerPC) このエラーはスタックフレームがないアセンブラ関数が原因です。そのような関数の場合、局所変数の制限は 224 バイトまでです。

解決方法 `fralloc` や `frfree` 疑似命令を使ってスタックフレームを作成してください。

local variable *name* was not assigned to a register

(Mac OS PowerPC) `register` 変数名がついているのに、対応するレジスタが宣言されていないとき、このエラーが発生します。アセンブラ関数では、`register` 変数はすべてレジスタに割り当てられ、その名前はレジスタが有効になった時点で使われます。

解決方法 これは既に多くの `register` 変数が使われているときに発生します。先に宣言した `register` 変数はずして、このエラーを解析してください。

M ~ O (C/C++)

M、N、O で始まるエラーメッセージについて説明します。

エラー 10009 macro *Macro* redefined

既に定義されたマクロを再定義しています (このメッセージはマクロ `<Macro>` が既に定義されているという意味です)。

解決方法 該当マクロを削除するか、名前を変更してください。

エラー 10012 macro(s) too complex

マクロ定義が複雑 (または再帰的) すぎて、展開できません。

解決方法 マクロを調べて、もっと単純なマクロを設計してください。

エラー 10235 'main' not defined as external 'int main()' function

現時点では文書にされていません。

エラー 10200 method *mthd* not defined

@ インターフェースで宣言されたメソッドの定義がなされていません。

エラー 10194 method *mthd* redeclared

既に宣言されたメソッド `mthd` を宣言しています。

エラー 10201 method *mthd* redefined

既に定義されたメソッド *mthd* を定義しています。

エラー 10237 name has not been declared in namespace/class

ネームスペースで宣言されていないものを定義しています。

```
namespace N {  
int N::a; // <<<error
```

エラー 10173 no parameters allowed in SOM class constructors

SOM クラスのコンストラクタ関数内にコンストラクタ関数を含めることはできません。SOM オブジェクトの詳細については『C Compilers Reference』または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10172 no static members allowed in SOM classes

SOM クラスは静的データメンバを含むことができません。SOM オブジェクトの詳細については『C Compilers Reference』または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10129 non-const '&' reference initialized to temporary

参照型の初期値が、その型の左辺値ではありません。このとき、コンパイラはテンポラリ変数を作成します。しかし、このテンポラリ変数はメモリにはありません ([リスト 2.90](#))。

参考文献 [『not an lvalue』\(p50\)](#)

リスト 2.90 構造体ではなく、かつ、参照がテンポラリ変数で初期化される例

```
long &r = 40000;  
// the proper method to use is  
long x;  
long &y = x;  
y = 40000;
```

エラー 10183 new SOM callstyle method *method* must have explicit \uparrow Environment * 1 parameter

新しい IDL の呼び出しスタイルを使って SOM クラスを作成するとき、各クラスのメソッドは環境ポインタを第 1 引数としなければなりません。SOM オブジェクトの詳細については『C Compilers Reference』または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 2 つの方法があります。

環境ポインタを第 1 引数としてメソッドのテーブルに追加します。

プリAGMA SOMCallStyle を使って、すべてのクラスメソッドが OIDL 呼び出しスタイルを使うようにします。SOMCallStyle メソッドを次に示します。

```
#pragma SOMCallStyle OIDL
```

エラー 10050 not a struct/union/class

構造体、共有体、クラスを使っている変数が、単純な型で再宣言されています。

リスト 2.91 構造体ではない

```
long var;  
var.myfoo = 10; // error: var は struct ではない
```

エラー 10043 not an lvalue

式が左辺値 (lvalue) ではありません。左辺値は代入式の左辺に書く式です。コンパイラは式が変数などのアイテムを参照し、値を代入できることを想定しています。

エラー 10055 number is out of range

数値が該当データ型の範囲を越えています。

参考文献 Metrowerks C/C++ でサポートされているデータ型については、『C Compilers Reference』を参照してください

エラー 10234 object *object* redefined

オブジェクトが不当に再定義されています ([リスト 2.92](#))。

リスト 2.92 オブジェクト *object* の再定義

```
void foo() {}  
void foo() {} //<<< ERROR
```

エラー 10198 Objective-C type *Type* is undefined (should be defined in objc.h)

objc.h ヘッダファイルで定義されていない Objective-C 型の *Type* を使っています。

エラー 10199 Objective-C type *Type* has unexpected type

予想外のオブジェクト型を含むタイプ *Type* を使っています。

P ~ R (C/C++)

P、Q、R で始まるエラーメッセージについて説明します。

エラー 10127 pascal function cannot be overloaded

CodeWarrior では pascal 関数のオーバーロードは許可されていません ([リスト 2.93](#))。

リスト 2.93 不当な pascal 関数の多重定義

```
int f(int);
pascal void f();    // error
```

エラー 10049 pointer/array required

[がポインタ名または配列名以外のところで使われています。[はポインタまたは配列名の後だけに書かれるトークンです。

エラー 10107 possible unwanted ';'

(警告メッセージ) while、if、for のステートメントで、';' が不要と思われるところに、';' が書かれています。C/C++ Warnings 設定パネルで『Possible Errors』オプションをオンにしている場合のみ、このエラーが発生します ([リスト 2.94](#))。

解決方法 不要な ';' を削除してください。または、(このステートメントでよいならば) Warnings 設定パネルで『Possible Errors』をオフにして、このエラーが出ないようにしてください。

参考文献 『Possible Errors』オプションについては『C Compilers Reference』の第2章を参照してください。

リスト 2.94 不要な ';'

```
while (x < 10); // 不要な ';'
printf("%d ", x);
```

エラー 10191 'pointer to member' is not supported for SOM classes

SOMObject を継承したクラスのメンバのアドレスを取得することはできません。例えば、foo が SOMObject を継承しているとき、&foo::bar は使用できません。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

エラー 10108 possible unwanted assignment

(警告メッセージ) while、if、for のステートメントで、等号比較命令が予想されるところに、代入命令が書かれています。このエラーは、C/C++ Warnings 設定パネルで『Possible Errors』オプションをオンにしている場合だけ発生します。

解決方法 この代入命令を正しい等号比較命令へ変更してください。または、(このステートメントでよいならば) C/C++ Warnings 設定パネルで『Possible Errors』をオフにして、このエラーが出ないようにしてください。

参考文献 『Possible Errors』 オプションについては 『C Compilers Reference』 の第 2 章を参照してください。

リスト 2.95 不要な代入文

```
if (x = 20) printf("OK"); // 不要な代入文
```

エラー 10109 possible unwanted compare

(警告) [リスト 2.96](#) のような等号比較命令がエラーになります。

注意： このエラーは、C/C++ Warnings 設定パネルで 『Possible Errors』 をオンにしている場合だけ発生します。

解決方法 この等号比較命令を正しい代入命令へ変更してください。または、(このステートメントでよいならば) C/C++ Warnings 設定パネルで 『Possible Errors』 をオフにして、このエラーが出ないようにしてください。

参考文献 『Possible Errors』 オプションについては 『C Compilers Reference』 の第 2 章を参照してください。

リスト 2.96 不要な等号比較命令

```
x == 1; // 不要な等号比較命令
```

エラー 10019 preceding #if is missing

#endif 疑似命令に対応する #if 疑似命令が前方にありません。

解決方法 前方にネストしている #if の構造体を調べ、余分な #endif 疑似命令がないかどうかを調べてください。

エラー 10138 preceding '#pragma push' is missing

#pragma pop 命令に対応する #pragma push 命令が前方にありません。

CodeWarrior の pragma 命令の詳細は 『C Compilers Reference』 を参照してください。

エラー 10219 preprocessor #error directive

#error 命令があります。

注意： このエラーを解決する前に、なぜこの命令が追加されたのかを調べてください。
#error 命令は通常、ある部分のコードのコンパイルを防ぐために使用されます。

解決方法 `#error` 命令を除去してください。

エラー 10238 `preprocessor #warning directive`

`#warning` 疑似命令があります。

注意： このエラーを解決する前に、なぜこの命令が追加されたのかを調べてください。
`#warning` 命令は通常、ある部分のコードの予期せぬ結果を知らせるために使用されま
す。

注意： C/C++ Language 設定パネルで『`』`がオンの場合、この疑似命令は使用できませ
ん。

解決方法 `#warning` 疑似命令を削除してください。

エラー 10018 `preprocessor syntax error`

不当なプリプロセッサ命令です ([リスト 2.97](#))。

解決方法 該当命令のシンタックスを調べてください。エラーを特定できないときは、エラー箇所よ
り前方でエラーを捜してください。

リスト 2.97 プリプロセッサシンタックスエラー

```
#include file
```

エラー 10209 `protocol prtcl redefined`

プロトコル `prtcl` は既に定義されています。

エラー 10211 `protocol prtcl is already in protocol list`

プロトコル `prtcl` は既にプロトコルリストにあります。

エラー 10210 `protocol prtcl is undefined`

プロトコル `prtcl` は定義されていません。

エラー 10225 `protocol list does not match @interface`

インターフェースで宣言されたプロトコルリストが実装と一致しません。

リスト 2.98 @ インターフェースとプロトコルリストが一致しない

```
@protocol a
@end
@protocol b
@end
@interface A <a>
@end
@implementation A <b> //<<< ERROR: 宣言と一致しない
@end
```

エラー 10205 receiver cannot handle this message

受信側がこのメッセージを正しく処理できません。

エラー 10061 reference to label `lbl` is out of range

アセンブラ関数の中に、参照範囲外のアドレスへの分岐命令があります ([リスト 2.99](#))。

リスト 2.99 参照範囲外にあるラベル '`lbl`' への参照

```
bra.s label// error: label が遠すぎる
```

エラー 10085 return value expected

戻り値を返すことが想定されている関数が、戻り値を返しません。例えば、[リスト 2.100](#) の `var()` は `int` 型の戻り値を返すか、関数を `void` 宣言にする必要があります。

解決方法 該当する関数を `void` 宣言するか、値を返すようにしてください。

警告! C++ では、戻り値なしで宣言された関数は `int` 型を返すことになっています ([リスト 2.101](#))。

リスト 2.100 想定される戻り値を返していない

```
int var() {} //error: 戻り値がない
```

リスト 2.101 `main()` 関数からの戻り値が必要

```
main()
{
    cout << "working";
```

```
        //<-- 戻り値がないエラー  
    }
```

エラー 10158 RTTI option is disabled

RTTI (Run Time Type Identification) が必要なときに、RTTI の `#pragma` が設定されていないとき、または C/C++ Language 設定パネルの『Enable RTTI』がオフのとき、このエラーが発生します。

解決方法 C/C++ Language 設定パネルの『Enable RTTI』をオンにしてください。

S ~ T (C/C++)

S、T で始まるエラーメッセージについて説明します。

エラー 10187 sizeof() is not supported for SOM classes

SOMObject を継承したオブジェクトまたはクラスを `sizeof()` で使うことはできません。SOM オブジェクトの詳細については『C Compilers Reference』または『SOMObjects Developer Toolkit (IBM)』をご覧ください。

エラー 10180 SOM class access qualification only allowed to direct parent or own class

明示的スコープ (`obj->B::func()` など) をともなってメソッドを起動するとき、特定のクラス (B) はオブジェクト (`obj`) と同じクラス、またはオブジェクトの直接の親クラスでなければなりません。

例えば、クラス A がクラス B (クラス B はクラス C の親クラス) の親クラスの場合

```
C* obj = new C;
```

```
obj->C::func(); // OK: C は obj のクラス
```

```
obj->B::func(); // OK: B は obj クラスの直接の親
```

```
obj->A::func(); // ERROR: A は obj クラスの直接の親ではない
```

SOM オブジェクトの詳細については『C Compilers Reference』または『SOMObjects Developer Toolkit (IBM)』をご覧ください。

エラー 10190 SOM class arrays are not supported

SOM オブジェクトの配列を作成することはできません。SOM オブジェクトの詳細については『C Compilers Reference』または『SOMObjects Developer Toolkit (IBM)』をご覧ください。

解決方法 SOM オブジェクトを配列以外の方法で保存してください (リンクリストなど)。

エラー 10170 SOM class data members must be private

SOM クラスのすべてのデータメンバは必ず非公開でアクセスしなければなりません。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 SOM のデータメンバを限定公開や公開で宣言しないようにしてください。

エラー 10188 SOM classes cannot be class members

SOM クラスを他のクラスのメンバとして宣言することはできません。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 SOM オブジェクトへのポインタをメンバ宣言してください。

エラー 10168 SOM classes can only inherit from other SOM based classes

SOM クラスは SOMObject を継承したクラスのみを継承します。多重継承を使っている場合、SOM クラスと普通のクラスを合わせて使うことはできません。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 すべての SOM クラスの基底クラスが SOMObject からの継承であることを確認してください。SOM クラスを作成しない場合、基底クラスが SOMObject を継承していないことを確認してください。

エラー 10169 SOM classes inheritance must be virtual

SOM クラスを宣言するとき、そのすべての基底クラスは仮想でなければなりません。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 `virtual` キーワードがクラス宣言の各基底クラスの前にあることを確認してください。

エラー 10192 SOM class has no release order list

(警告) `SOMReleaseOrder` リストなしで SOM クラスを作成し、かつ『Extended Error Checking』がオンのとき、このエラーが発生します。コンパイラはリリースオーダーがクラス宣言で関数が出てくる順番と同じであることを想定しています。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 この警告を避ける方法は 2 つあります。

C/C++ Warnings 設定パネルで『Extended Error Checking』をオフにします。

プラグマ `SOMReleaseOrder` を使ってクラスのメンバ関数のリリースオーダーを指定します。

エラー 10181 SOM class must have one non-inline member function

SOM クラスはインライン関数ではないメンバ関数を少なくとも 1 つ持たなければなりません。MacSOM はどのトランスレーションユニットがクラスをインプリメントするかを決めるためにこのクラスを使います。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 SOM クラスはインライン関数ではないメンバ関数を少なくとも 1 つ持つことを確認してください。必要ならば、空のメンバ関数を作成してください。

エラー 10175 SOM runtime function *func* not defined (should be defined in somobj.hh)

コンパイラはランタイム関数が `somobj.hh` ヘッダにあることを想定していますが、それを発見できません。`somobj.hh` ファイルが壊れているか、そのファイルの内容が正しくない可能性があります。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 ハードディスクに既にインストールされている `somobj.hh` ヘッダファイルを CodeWarrior CD のものにへ入れ替えてください。MacSOM を熟知している方でしたら、`somobj.hh` を変更してもよいでしょう。

エラー 10176 SOM runtime function *func* has unexpected type

コンパイラはランタイム関数が `somobj.hh` ヘッダに定義されていることを想定していますが、リターン型が正しくありません。`somobj.hh` ファイルが壊れているか、そのファイルの内容が正しくない可能性があります。SOM オブジェクトの詳細については『C Compilers Reference』、または「SOMObjects Developer Toolkit (IBM)」をご覧ください。

解決方法 ハードディスクに既にインストールされている `somobj.hh` ヘッダファイルを CodeWarrior CD のものに入れ替えてください。MacSOM を熟知している方でしたら、`somobj.hh` を変更してもよいでしょう。

エラー 10182 SOM type *variable* undefined

特定の SOM データ型 (Environment など) がありません。正しいヘッダファイルをインクルードしていないことが原因であることが多いです。

エラー 10226 super class does not match @interface

インターフェースで定義されているスーパークラスと実装が一致しません。

リスト 2.102 @ インターフェースとスーパークラスが一致しない

```
@interface a
@end
@interface b
@end
@interface c : a
@end
```

```
@implementation c : b//<<< ERROR: 宣言と一致しない
@end
```

エラー 10007 string too long

文字列が長すぎます。

解決方法 通常、文字列の最後に終了を示すマークがないためこのエラーが発生します。このような部分を見つけやすくするために、Editor Settings 環境設定パネルの『Color Syntax』をオンにしてください。

エラー 10034 struct/union/class member *stType* redefined

既に定義されている構造体、共有体、列挙型、クラスメンバを再定義してます。

解決方法 該当する構造体、共有体、列挙型、クラスメンバを削除するか、名前を変更してください。

エラー 10038 struct/union/class size exceeds 32k

構造体、共有体、クラスのサイズが 32k を越えています。構造体、共有体、クラスは通常上限 32k のスタックに格納されます。

解決方法 配列を静的に定義して制限をなくす、または動的割り当てを行ってスタックではなくヒープに格納することができます。

エラー 10033 struct/union/enum/class tag *stType* redefined

既に定義されている構造体、共有体、列挙型、クラスタグを再定義してます。

解決方法 該当する構造体、共有体、列挙型、クラスタグを削除するか、名前を変更してください。

エラー 10135 template redefined

既に定義されているテンプレートを再定義してます ([リスト 2.103](#))。

解決方法 該当するテンプレートを削除するか、名前を変更してください。

リスト 2.103 テンプレートを再定義している

```
template <class T> class aClass { ... };

template <class T> class aClass { ... }; // error
```

エラー 10136 template parameter mismatch

メンバ関数のテンプレート引数リストが、クラスの引数リストと一致しません ([リスト 2.104](#))。

リスト 2.104 テンプレート引数が一致しない

```
template <class T> class aClass { void f() };
template <class T,class U>
    void aClass<T>::f() { ... }; // error
```

エラー 10215 template too complex or recursive

リカーシブなテンプレート拡張が多すぎます。

解決方法 アルゴリズムを単純にしてください。

エラー 10052 the file *filename* cannot be opened

#include 命令で指定されたファイルが見つかりません(例えば、[リスト 2.105](#)で Foo.h が
見つかりません)。

解決方法 インクルードファイル名の入力を間違う、または、アクセスパスが正しくない可能性があります。『Find』コマンドでインクルードファイルがあることを確認してください。

または、アクセスパスの設定で、"... "で指定するところを "<...>" を指定しているか
もしれません。この場合は、Access Path 設定パネルの『Always Search User Paths』をオンに
するとエラーがなくなります。

参考文献 Access Path 設定パネルの『Always Search User Paths』の詳細は『C Compilers Reference』を
ご覧ください。

リスト 2.105 インクルードファイルを開けない

```
#include "AbstractHeader.h"
#include "Foo.h" // このファイルは存在しない
#include <QDoffscreen.h>
```

エラー 10167 the parameter(s) of the *functionname* function must be immediate value(s)

現時点では文書にされていません。

エラー 10048 too many initializers

宣言されているアイテム数よりも多い初期値を初期化構造体に渡しました。

解決方法 配列、構造体、クラスの初期化において、実際のエレメント数より多い数を宣言しようと
したとき、このエラーが発生します。エレメントの数をあわせて、正しい個数分だけ値を
宣言するようにしてください。

エラー 10011 too many macro arguments

32 個以上の引数を持つマクロ定義があります ([リスト 2.106](#))。

リスト 2.106 マクロ定義の引数が多すぎる

```
#define macro(arg1,...,arg33) ...
```

エラー 10146 type mismatch *A_type* and *B_type*

予想外のデータ型が使われました。

または、ユーザーが定義した関数が Metrowerks の提供するマクロ名と同じです ([リスト 2.107](#))。

解決方法 マクロがキャスト可能なデータ型であるとき、そのマクロで定義されたタイプキャストがデータに対して使われます。

[リスト 2.107](#) では、Length が Metrowerks のプリコンパイルヘッダ (MacHeaders68K および Mac HeadersPPC) の中で、以下のようにマクロ定義されているためにエラーとなります。

```
#define Length(s) (*(unsigned char *)(s))
```

このエラーを避けるために、自分の変数名または関数名を違う名前に変更してください。

リスト 2.107 型が一致しない

```
class MyLine {  
public:  
    MLine(double theLen);  
    double Length(void);
```

エラー 10233 typename redefined

型名が再定義されています ([リスト 2.108](#))。

リスト 2.108 型名の再定義

```
struct B;  
typedef int B;//<<<
```

U ~ Z (C/C++)

U、V、W、X、Y、Z で始まるエラーメッセージについて説明します。

エラー 10195 undefined method *mthd*

定義されていないメソッド名 *mthd* が使われています。

エラー 10041 undefined identifier *var*

定義されていない識別子が使われています。

解決方法 変数のスコープ範囲内で変数が宣言されているかどうかを確認してください。または、スペルミスかもしれません。

エラー 10060 undefined label *Lb1*

関数内で定義されていないラベルへの `goto` 命令があります。

解決方法 `goto` ステートメントを削除してください。または、エラーが発生したスコープ範囲内に、必要なラベルを設定してください。

エラー 10005 undefined preprocessor directive

Metrowerks C/C++ コンパイラが該当プリプロセッサ命令を認識できません。

解決方法 Metrowerks C/C++ コンパイラで認識できるプリプロセッサ命令の一覧は『C Compilers Reference』に記述されています。使用しているプリプロセッサ命令を、この文書で確認してください。

エラー 10003 unexpected end of file

文の内容が完結していないのに、ファイルの終わりに来ました。

解決方法 このエラーは、スペースを忘れたり、} (右ブレース) がバランスしていなかったりといったことが原因で発生します。問題のあったソースコードの行を調べてください。エラーが特定できないときは、前方にあるエラーを修正して再コンパイルしてください。

エラー 10013 unexpected end of line

文の内容が完結せずに行の終わりに来ました。

解決方法 このエラーは、';' を入力し忘れたり、シンボルを書き忘れたことが原因で発生します。問題のあったソースコードの行を調べてください。エラーが特定できないときは、前方にあるエラーを修正して再コンパイルしてください。

エラー 10021 unexpected token

コンパイラが予想しないところにトークンがあります。

解決方法 エラーが特定できないときは、前方に何か問題があります。前方にあるシンタックスエラーおよび書き忘れたシンボルを修正して再コンパイルしてください。

エラー 10091 unimplemented C++ feature

Metrowerks C++ コンパイラでサポートされていないC++ の機能が使われました。

エラー 10162 unknown assembler instruction mnemonic

不当な命令名があります。

解決方法 アセンブラ命令のニーモニックを修正してください。

エラー 10207 unknown message selector

オブジェクト階層でセレクトが宣言または定義されていません。

エラー 10020 unterminated #if / macro

#if 命令に対応する #endif 命令がありません。または、マクロ定義が不完全です。

エラー 10004 unterminated comment

コメント文を完了しないでソースコードが終了しています。

エラー 10068 value is not stored in register

現時点では文書にされていません。

エラー 10086 variable *var* is not initialized before being used

変数へ代入や、変数を初期化をする前に、該当変数を使おうとしています ([リスト 2.109](#))。

解決方法 変数を使う前に、変数を初期化してください。

リスト 2.109 変数が初期化されていない

```
static int f()  
{  
    int i;  
    return i;  
}
```

エラー 10083 variable *var* is not used in function

(警告) 関数内で宣言されている変数が、関数内で使われていません。C/C++ Warnings 設定パネルの『Unused Variables』がオンのときに、このようなエラーを見つけることができます。

解決方法 使われていない変数を削除するか、C/C++ Warnings 設定パネルの『Unused Variables』をオフにしてください。

参考文献 『Unused Variables』の詳細は『C Compilers Reference』をご覧ください。

エラー 10120 仮想関数を `pascal` 宣言しています。 [リスト 2.110](#) では、クラス `var` で `func()` を `pascal` 関数として宣言しています。

リスト 2.110 pascal 関数を仮想関数にはできない

```
class aClass {  
    pascal int func(); // 不当な宣言  
};
```

索引

ア行

エラー

C/C++ コンパイラ 7

フォントの説明 5

カ行

コンパイラのエラー

C/C++ 68k 7

C/C++ PPC 7

Win32/x86 7

サ行

説明

フォント 5

CodeWarrior

Error Reference

Credits

writing lead: John Roseborough

other writers: Ron Liechty, Marc Paquette

engineering: Marcel Achim, Mark Anderson, Berardino Barrata,
Pascal Cleve, Michael Cordery, Andreas Hommel,
Nick Havens, John McEnerney, Clinton Popetz,
Cam Vien

frontline warriors: Metrowerks Technical Support

translation: Naomi Owashi

CodeWarrior 文書のガイド

CodeWarrior の文書はツールと同様にモジュールのように構成されています。ツール、言語、ライブラリ、ターゲットごとにマニュアルがあります。各 CodeWarrior 製品によって含まれるマニュアルが異なります。この表に記載されていないマニュアルが含まれることもあります。

コアマニュアル	
IDE User Guide	CodeWarrior IDE の使い方
Debugger User Guide	CodeWarrior デバッガの使い方
CodeWarrior Core Tutorials	IDE の各ツールのチュートリアル
言語 / コンパイラのマニュアル	
C Compilers Reference	C/C++ フロントエンドコンパイラの情報
Pascal Compilers Reference	Pascal フロントエンドコンパイラの情報
Error Reference	コンパイラ / リンカのエラーメッセージのリストおよび解説
Pascal Language Reference	Metrowerks における ANS Pascal の実装
Assembler Guide	スタンドアローンアセンブラのシンタックス
Command-Line Tools Reference	Mac OS / BeOS コンパイラのコマンドラインのオプション
Plugin API Manual	CodeWarrior のプラグインコンパイラ / リンカの API
ライブラリのマニュアル	
MSL C Reference	Metrowerks ANSI 標準 C ライブラリの関数のリファレンス
MSL C++ Reference	Metrowerks ANSI 標準 C++ ライブラリの関数のリファレンス
Pascal Library Reference	Metrowerks ANS Pascal ライブラリの関数のリファレンス
MFC Reference	Win32 用の Microsoft Foundation Classes リファレンス
Win32 SDK Reference	Win32 API の Microsoft リファレンス
The PowerPlant Book	Mac OS 用アプリケーションフレームワークのガイド
PowerPlant Advanced Topics	PowerPlant での Mac OS プログラミングの高度なテクニック
ターゲットマニュアル	
Targeting BeOS	BeOS プログラミングでの CodeWarrior の使い方
Targeting Java VM	Java 仮想マシンプログラミングでの CodeWarrior の使い方
Targeting Mac OS	Mac OS プログラミングでの CodeWarrior の使い方
Targeting MIPS	MIPS 組み込みプロセッサプログラミングでの CodeWarrior の使い方
Targeting NEC V800 series	NEC V810/830 プロセッサプログラミングでの CodeWarrior の使い方
Targeting Net Yaroze	Net Yaroze プログラミングでの CodeWarrior の使い方
Targeting Palm OS	Palm OS プログラミングでの CodeWarrior の使い方
Targeting PlayStation OS	PlayStation プログラミングでの CodeWarrior の使い方
Targeting PowerPC	PPC 組み込みプロセッサプログラミングでの CodeWarrior の使い方
Targeting Win32	Windows プログラミングでの CodeWarrior の使い方

印の付いているマニュアルは日本語訳が用意されています。