14. C, C++に関する知識 II

1. 科目の概要

C 言語を拡張してオブジェクト指向の概念を取り入れたプログラミング言語である C++の基本的な知識を解説する。C++プログラムの構造、型、演算子や標準的なプログラム記述方法、ライブラリを利用したプログラミングなどについて説明する。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説 明	シラバスの対応コマ
II-14-1. C++言語の歴史と特徴、開発事例と開発手順	C++の歴史や特徴、C++による開発事例を解説する。また、C++によるプログラムを開発する手順として、エディタによるプログラム作成からコンパイル、プログラム実行までの流れを説明、簡単なC++プログラム作成例を紹介する。	10
II-14-2. C++プログラムの構成、Cとの相違点	C++プログラム構成の概要を説明し、とくにC++言語と比較した際の相違点や新たに導入された特徴について解説する。オブジェクト指向とは何かを解説し、C++で導入されているオブジェクト指向の特徴について述べる。	10
II-14-3. 基本的なC++プログラムの記述方法	C++プログラムの基本的な記述方法やプログラム構成を説明し、変数、データ型、メモリ管理、関数、標準ライブラリ、例外処理など、とくにC言語から拡張された部分を中心としてC++プログラミングに必要な基本的要素について解説する。	11
II-14-4a. オブジェクト指向の概念	C++プログラミングの前提となるオブジェクト指向の概念について解説する。データ自身だけでなくデータの処理方法まで含めてオブジェクトに全てが属するという根本的な考え方と、各種の用語、型による抽象的なデータ操作などについて説明する。	12
II-14-4. C++言語によるオブジェクト指向プログラミング	C++によるオブジェクト指向プログラミングの実現方法について解説する。C++におけるクラスの定義方法、コンストラクタ、デストラクタ、クラスの継承、オーバーライド、オーバロード、多重継承などについて説明する。	12
II-14-5. Standard Template Library	コンテナ、イテレータ、アルゴリズムのライブラリ化といった概念を説明し、C++によるその実装であるSTL (Standard Template Library)を紹介する。またSTLによるプログラミング例を挙げ、STLの具体的な使い方を解説する。	13
(削除) II-14-6. STLを利用したプログラミング	(上記とマージした) STLによるプログラミング例を挙げ、STLの具体的な使い方を解説する。線形リストやその他のデータ構造をSTLでどう利用できるかを示し、STLを利用したプログラミングのコツや注意すべきポイントなどを示す。	13
II-14-7. GTK+によるGUIプログラミング	GUIアプリケーションを開発するために用意されたウィジェットライブラリであるGTK+を紹介する。GTK+の導入方法、基本的な構造、特徴について述べ、GTK+で使われるシグナルとコールバックの概念を説明する。また簡単なサンプルプログラムによりGTK+の使い方を示す。	14
II-14-8. QtによるGUIプログラミング	GUIアプリケーションを開発するために用意されたもうひとつの著名なウィジェットライブラリであるQtを紹介する。Qtの導入方法、基本的な構造、特徴について述べ、Qtで使われるシグナルとスロットの概念を説明する。また簡単なサンプルプログラムによりQtの使い方を示す。	14
II-14-9. ライブラリの利用例1 (DBライブラリ)	そもそもライブラリとは何かを説明し、ライブラリの活用例としてデータベースライブラリの利用方法を紹介する。ライブラリを介してデータベースを操作する方法、データの出し入れを行うためのサンプルプログラムを示し、データベースライブラリの使い方を解説する。	15
II-14-10. ライブラリの利用例2 (オプション解析ライブラリ)	もうひとつのライブラリ利用例として、コマンドラインオプションを解析するライブラリの使い方を紹介する。コマンドラインオプションとは何か、どのようなスタイルがあるかを説明し、コマンドラインオプション解析ライブラリを用いてプログラムからコマンドラインオプションを解析するやり方を説明する。	15

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「14. C, C++に関する知識Ⅱ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル（Ⅰ）									応用レベル（Ⅱ）					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14. C、C++に関するスキル	<Cの基本>	<Cの基本構造>	<文字列操作>	<関数>	<ポインタ>	<構造体>	<コンソール入出力>	<ファイル管理>	<データ構造>	<C++の基本>	<C++の基本構造>	<オブジェクト指向プログラミング>	<STL（Standard Template Library）>	<GUIアプリケーションの開発>	<開発ライブラリの使用>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_14.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造 [14-1, 2, 3, 4, 5, 6, 9]	IT-PF2. プログラミングの基本的構成要素 [14-10]	IT-PF3. オブジェクト指向プログラミング [14-10, 12]	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 結合的コーディング [14-10, 12]	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要 [14-1, 2, 4] [14-10, 12]						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャとハードウェア	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CA00. 歴史と概要	CE-CA01. コンピュータアーキテクチャの基礎	CE-CA02. メモリシステムの構成とアーキテクチャ	CE-CA03. インタフェースと通信	CE-CA04. デバイスサブシステム	CE-CA05. CPUアーキテクチャ	CE-CA06. 性能・コスト評価	CE-CA07. 分散・並列処理	CE-CA08. コンピュータによる計算	CE-CA09. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア、	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、C++言語を用いた開発手法の知識がある。STL、GTK+、Qt などを用いたプログラミングを通して実践的な知識を習得する。

科目名	第10回	第11回	第12回	第13回	第14回	第15回
14.C、C++に関する知識Ⅱ	(1)C++の説明 (2)オブジェクト指向の説明 (3)C++による開発の流れ	(1)C++の書き方の特徴 (2)基本的なプログラム記述の例	(1)クラスの説明 (2)クラスの継承 (3)オブジェクト指向プログラミングの説明	(1)STL の説明 (2)STL によるプログラミング	(1)GTK+によるGUI アプリケーション開 (2)Qt によるGUI アプリケーション開	(1)開発ライブラリの説明 (2)データベースライブラリの説明 (3)コマンドラインオプションの解析ライブ

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-1. C++言語の歴史と特徴、開発事例と開発手順	
対応する コースウェア	第 10 回 C++の基本	

II-14-1. C++言語の歴史と特徴、開発事例と開発手順

C++の歴史や特徴、C++による開発事例を解説する。また、C++によるプログラムを開発する手順として、エディタによるプログラム作成からコンパイル、プログラム実行までの流れを説明、簡単な C++プログラム作成例を紹介する。

【学習の要点】

* C++言語は AT&T ベル研究所の Bjarne Stroustrup 博士が、事象駆動型のシミュレーションを記述するため C 言語を拡張した言語として開発した。
* C 言語にオブジェクト指向の概念を取り入れクラスと派生クラス、仮想関数と、関数と演算子の多重定義、参照型、const 型等の機能が追加された。
* オープンソースソフトウェアの C++言語コンパイラとしては gcc(g++)がある。

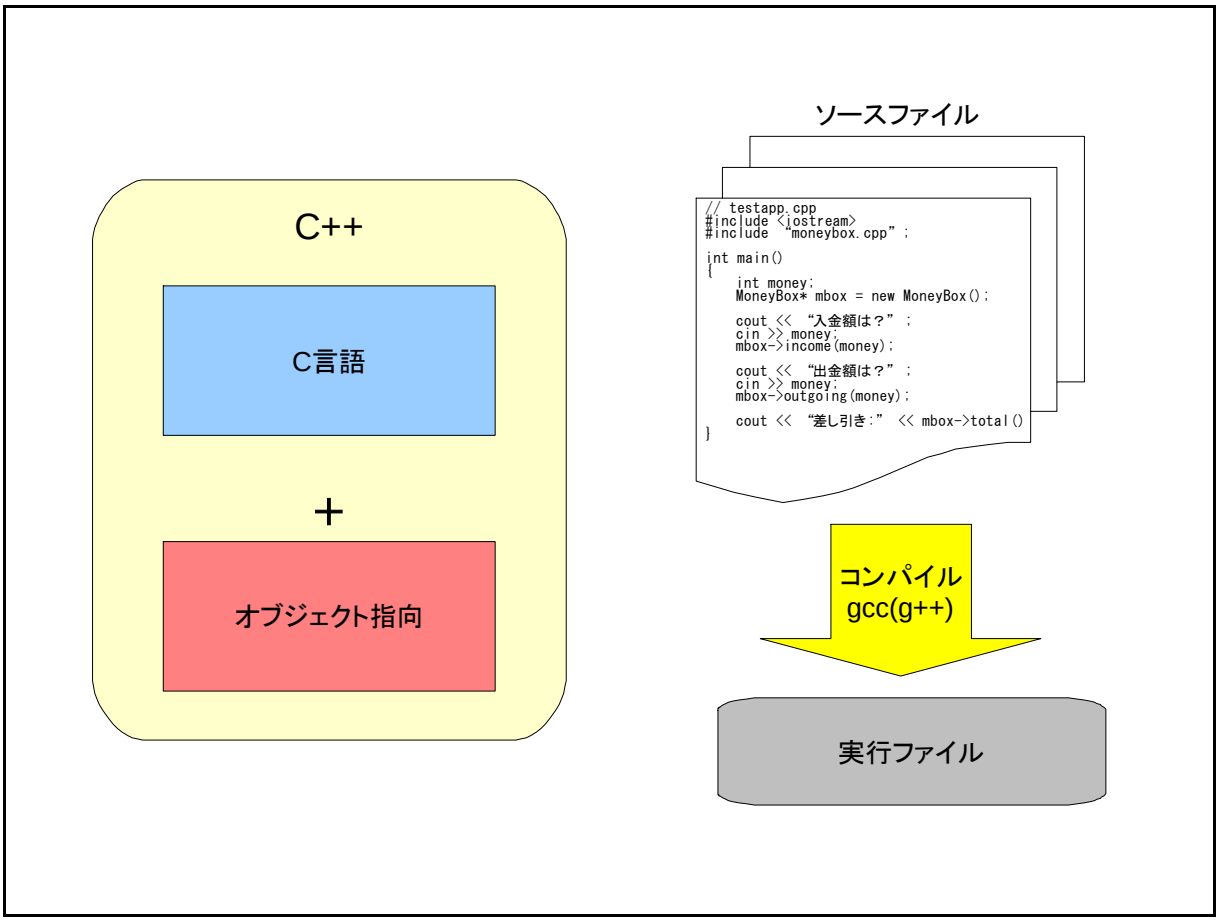

図 II-14-1.　C++のプログラミング

【解説】

1) C++の誕生

C 言語を始めこれまでのプログラム言語は、最初にデータの定義を行いそのデータを処理する手続きを記述する手続き型の言語であったため、現実の事象を表そうとしたときにデータの定義や手続きの記述が複雑になってしまう問題があった。
関連するデータとこれを用いる処理を1つのグループにまとめたオブジェクトにすることにより、現実の事象を表した理解しやすいプログラムを記述できると考え、オブジェクト指向の概念が生まれた。
C++は AT&T ベル研究所の Bjarne Stroustrup 博士が、事象駆動型のシミュレーションを記述するため C 言語にオブジェクトを定義するクラスを始めとした新たな機能を追加し改良を行った言語である。
C++は C 言語で開発されたプログラムを利用でき、他の言語よりも効率よく処理を行えることから急速に普及した。

2) C++の特徴

C++の主な特徴としては以下のような点が挙げられる。

＊ クラスと派生クラス
C++の最大の特徴としては、オブジェクト指向の概念であるデータと命令をクラス内で組み合わせるという機能が追加されたことである。また、C++ではこの概念をさらに発展させ、新しいクラスを作成する際に既存のクラスを継承した派生クラスとして作成できるようにしている。

＊ 仮想関数
元となるクラスにて関数を仮想関数として定義することにより、変数の型に関連する関数ではなく、その変数に格納されたオブジェクトの実態に関連する関数が実行されるようになる。

＊ 関数と演算子の多重定義
同じ関数名や演算子で複数の定義を行うことができるため、関数や演算子で扱う変数の型が違っていても同じ関数や演算子を用いて処理を行うことができる。

＊ 参照型
定義済みの変数を用いて参照型の変数を定義することにより、変数の別名として使用することができる。関数の引数として指定した場合、呼出し元で定義された変数を関数内で変更することが可能となる。

＊ const 型
const 型で変数を定義することにより、プログラム内で変更が出来ない変数として定義する。

3) C++を利用したプログラム開発

C++は C 言語と同様にソースコードをコンパイルして実行可能なファイルを作成することによりプログラムを作成する。オープンソースソフトウェアの C++コンパイラとしては gcc が利用できる。
Linux における C++プログラムの作成手順としては以下のような流れになる。

- テキストエディタでソースコードを記述
- gcc(g++)にてソースコードのコンパイルとリンクを行い、実行ファイルを作成
- 実行ファイルの実行

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-2. C++プログラムの構成、C との相違点	
対応する コースウェア	第 10 回 C++の基本	

II-14-2. C++プログラムの構成、C との相違点

C++プログラム構成の概要を説明し、とくに C 言語と比較した際の相違点や新たに導入された特徴について解説する。オブジェクト指向とは何かを解説し、C++で導入されているオブジェクト指向の特徴について述べる。

【学習の要点】

* C++プログラムはヘッダファイルの読込やクラスの定義を行う宣言部分と、main 関数やその他の関数を記述する関数部分に分かれる。
* オブジェクト指向へ対応するため、構造体にそのデータを扱うための関数をあわせたクラスの定義を行う。

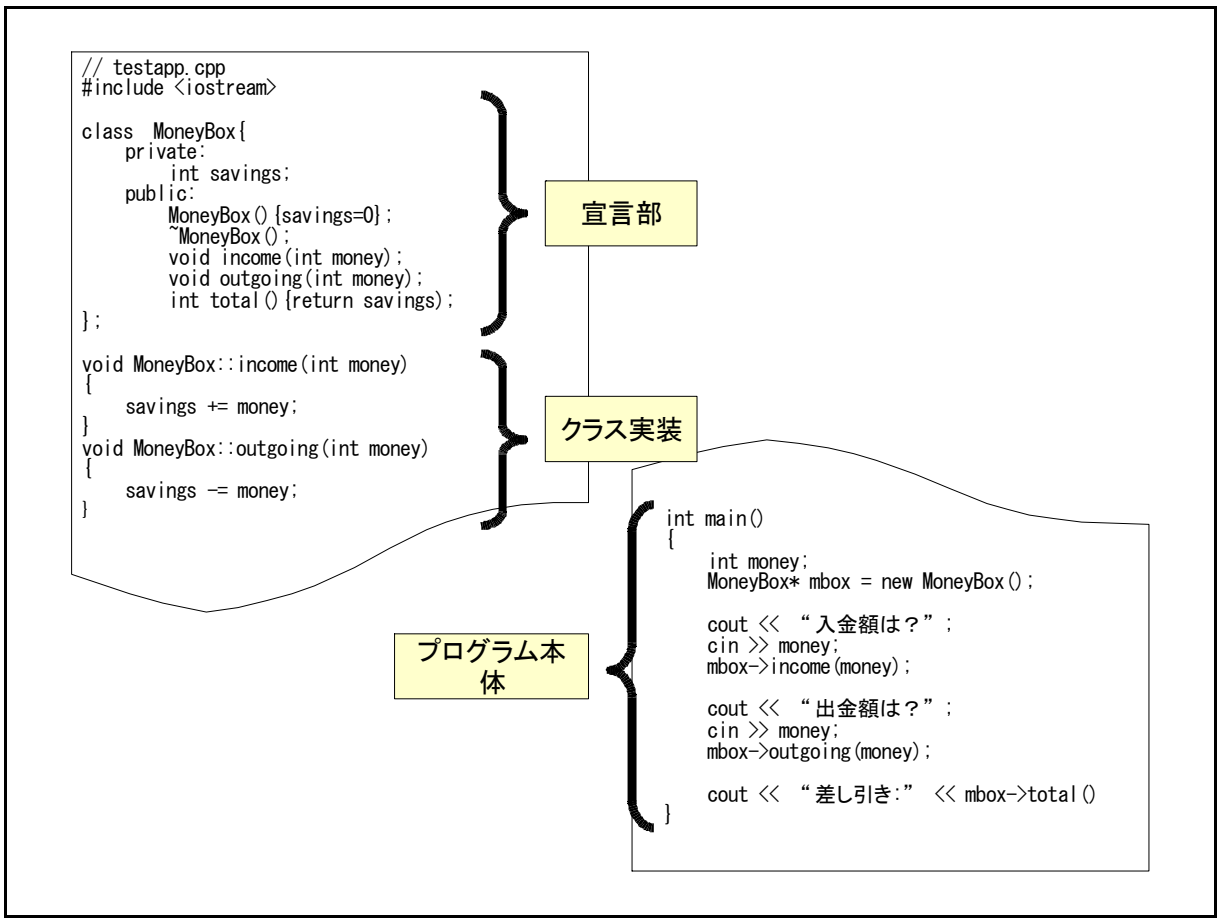

図 II-14-2.　C++プログラムの構成

【解説】

1) C++の基本構成

オブジェクト指向プログラミングは、関連のあるデータと処理をひとまとめにして1つのオブジェクトとして扱うプログラミング手法である。
C++ではデータを保存するメンバ変数と処理を実装するメンバ関数で構成されるクラスを使ってオブジェクト指向プログラミングを実現している。
C++はC 言語を元に開発されているため、基本的なプログラム構成はC 言語と同様の記述もできるが、オブジェクト指向プログラミングへ対応するため、以下のような3つの構成に分けてプログラムの記述を行う。

* 宣言部
  インクルードするファイルやヘッダファイル、クラスやインターフェイスの宣言を行う。
* クラスの実装
  クラスのメンバ関数の実体を記述する。
* プログラム本体
  クラスを利用して処理を行うプログラムを記述する。C 言語と同様に main 関数からプログラムが実行される。

2) 構造体とクラス

C 言語では関連するデータをまとめて管理する場合、以下のように構造体の定義によって実装を行い、この構造体を操作する処理は個別に関数の作成を行う。

* 記述例

```
struct stack {
    int count;
    int data[100];
}
void function func1(struct stack&, int item){
  …
}
```

C++では構造体と個別の関数で実装していたものを、以下のようにクラスとしてひとまとめにして実装することができる。

* 記述例

```
class stack {
    private:
      int count;
      int data[100];
    pubulic:
      void func1(int item);
};
void stack::func1(int item){
  …
}
```

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-3. 基本的な C++プログラムの記述方法	
対応する コースウェア	第 11 回 C++の基本構造	

II-14-3. 基本的な C++プログラムの記述方法

C++プログラムの基本的な記述方法やプログラム構成を説明し、変数、データ型、メモリ管理、関数、標準ライブラリ、例外処理など、とくに C 言語から拡張された部分を中心として C++プログラミングに必要な基本的要素について解説する。

【学習の要点】

- C++では変数をプログラム内のどこでも宣言することができる。宣言した変数のスコープは宣言したブロック内となる。
- 関数の引数を省略した場合、自動的に void 型となる。また関数定義またはプロトタイプ宣言にてデフォルト引数の指定ができる。
- C++ではプログラムの実行中にエラーが発生した場合、例外オブジェクトが生成されるためこれを利用してエラー処理を行う。

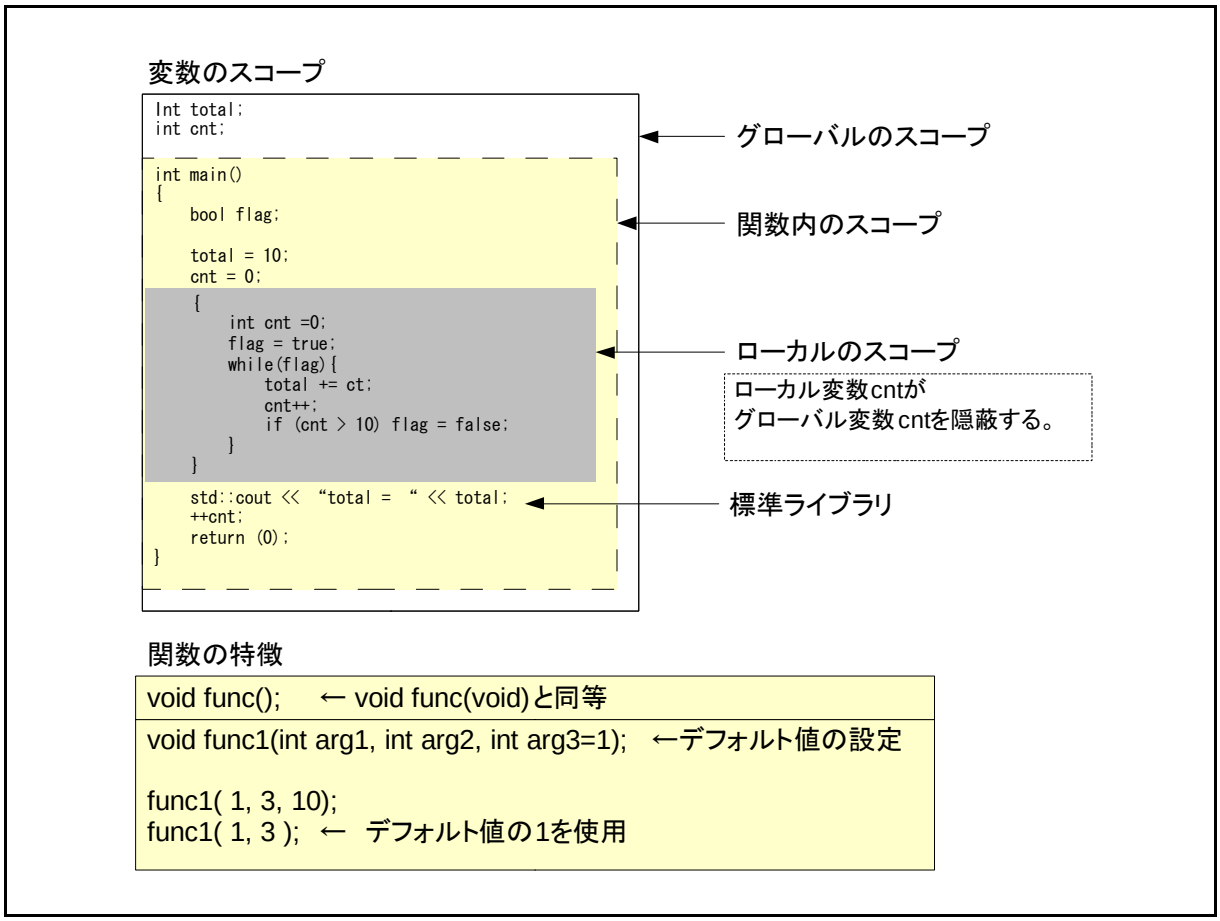

図 II-14-3. C++プログラムの記述

【解説】

以下に C++プログラム記述の主な特徴を示す。

1) 変数と定数

＊ 新しいデータ型

C++では、true または false の値を持つ「bool 型」が追加された。

＊ 変数の宣言とスコープ

C++では変数をプログラム内のどこでも宣言することができる。宣言した変数のスコープは宣言したブロック内となる。

＊ 名前空間

変数名が有効である範囲を名前空間として定義する。名前空間の定義は namespace 文で行う。定義済の名前空間に属する変数名は「名前空間名::変数名」のように指定して使用する。

2) 関数

＊ 引数のない関数

プロトタイプ宣言にて引数を持たない関数を定義する場合、C 言語では「(void)」を用いて宣言を行う必要があるが、C++では空の引数リスト「()」を用いて宣言を行うことができる。

＊ デフォルト引数

関数定義またはプロトタイプ宣言にて引数のデフォルト値を定義すると、関数の呼出し時に引数を省略することができる。引数を省略した場合デフォルト値が指定されたとして処理を行う。

＊ 関数テンプレート

template 宣言により、関数のテンプレートとなる処理を定義し、これを使用して実際に処理を行う関数の宣言を記述する。これにより、扱う引数の型が違う同じ関数名の関数を定義することができる。

3) 標準ライブラリ

C++で提供されるクラスでよく利用するものとしては以下のようなものがある。

＊ IOStream クラス

標準入出力へ読み書きを行うための一連のオブジェクトを提供する。

＊ string クラス

文字列の保存や演算などの機能を提供する。

＊ exception クラス

統一された例外処理の方法を提供する。

4) メモリ管理

オブジェクトの作成などメモリ領域を確保する時には、new 演算子を使用する。new 演算子にて確保したメモリ領域は、delete 演算子で開放する。

5) 例外処理

C++では try ブロック内に記述した処理で例外が発生した場合、catch 文で捕捉することができる。捕捉する例外は予めクラスとして定義しておき、catch 文でこのクラスを指定する。例外が発生すると、指定したクラスで例外オブジェクトが生成される。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-4. オブジェクト指向の概念	
対応する コースウェア	第 12 回オブジェクト指向プログラミング	

II-14-4. オブジェクト指向の概念

C++プログラミングの前提となるオブジェクト指向の概念について解説する。データ自身だけでなくデータの処理方法まで含めてオブジェクトに全てが属するという根本的な考え方と、各種の用語、型による抽象的なデータ操作などについて説明する。

【学習の要点】

＊ オブジェクトとは「属性」「操作」「関係」「アイデンティティ」の特性を満たしたものである。

＊ 同じ特性を持つオブジェクトの集合を抽象化したものを「クラス」といい、クラスに属するオブジェクトの例を「インスタンス」という。

＊ オブジェクト指向の基本的な技術としては「カプセル化」「継承」「多態性」がある。

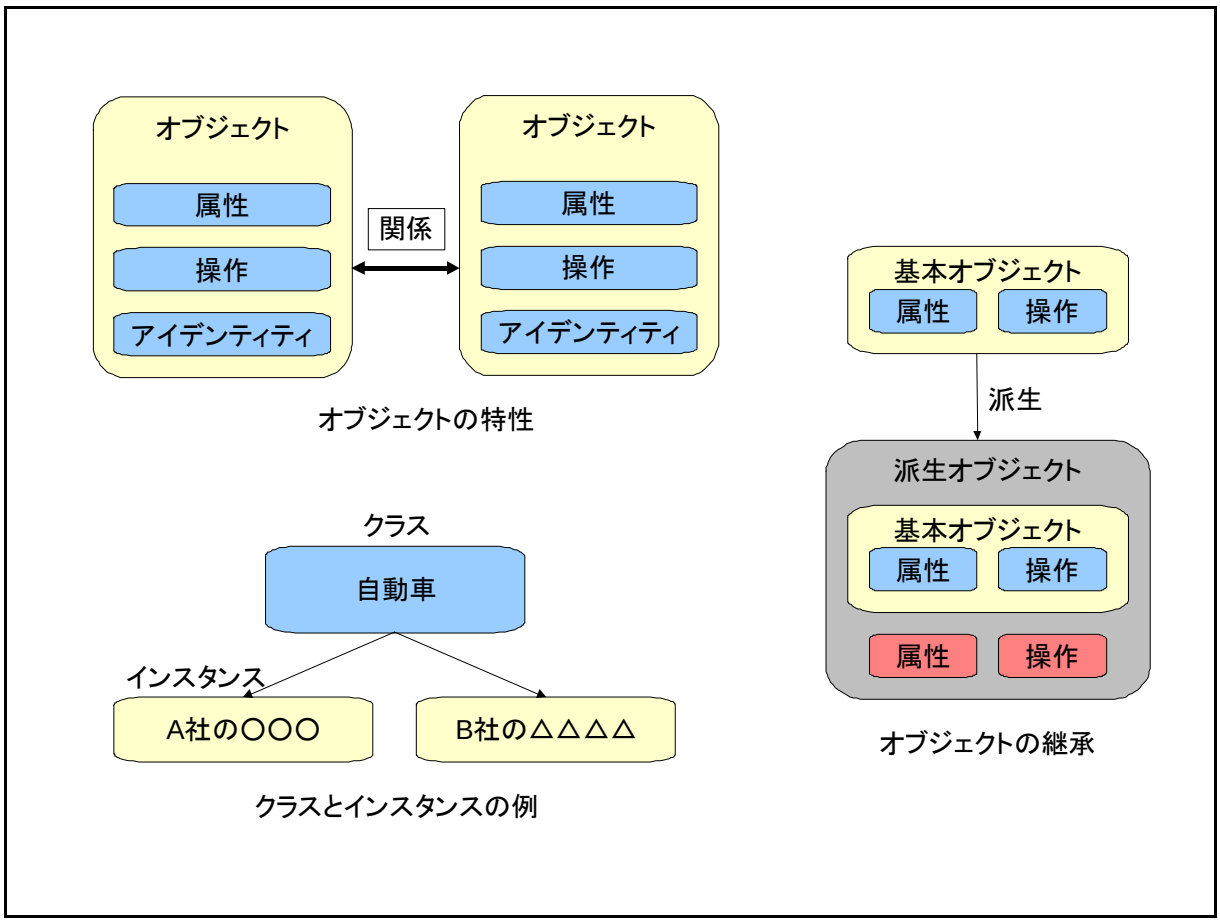

図 II-14-4.　オブジェクト指向の概念

【解説】

1) オブジェクトとは

オブジェクトとは「属性」「操作」「関係」「アイデンティティ」の特性を満たしたものである。

＊ 属性
オブジェクトは固有の形や性質などを持っている。

＊ 操作
オブジェクトは固有の振る舞いを持っている。

＊ 関係
オブジェクトは単独で存在することはなく、必ず他のものと関係を持っている。

＊ アイデンティティ
オブジェクトは他のものと区別するためのアイデンティティを持っている。

2) クラスとインスタンス

オブジェクトに関連する概念として「クラス」と「インスタンス」がある。

＊ クラス
オブジェクトの属性や操作の雛形を定義したものをクラスといい、クラスの定義はオブジェクトの集合である内包クラスとオブジェクトの抽象化である外延クラスの2つに分けられる。

＊ インスタンス
クラスから生成されるオブジェクトの実体のことをインスタンスと呼ぶ。

3) オブジェクト指向

オブジェクト指向では、実現する機能を、個々に属性と操作を持った「オブジェクト」として実装する。
またオブジェクト指向開発とは、オブジェクトの集まりを基本に開発を進める開発手法である。
オブジェクト指向の基本的な要素としては「カプセル化」「継承」「多態性」がある。

＊ カプセル化
オブジェクト同士の関係動作を、メッセージのみで行い、属性の詳細を隠ぺいすることを、カプセル化といい、他のオブジェクトからの変更を防ぐと共に、オブジェクト内部に変更があった場合でも他のオブジェクトへの影響を抑えることが可能となる。

＊ 継承
基本オブジェクトの特性を引き継いで派生オブジェクトを定義することを継承という。基本となるオブジェクトから継承したオブジェクトは新しい属性や操作を持つことができる。

＊ 多様性
基本オブジェクトで定義されている操作を、派生オブジェクトが持っている属性の型に応じて異なる処理が行われるように操作の実装を変化させることを多様性という。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-5. C++言語によるオブジェクト指向プログラミング	
対応する コースウェア	第 12 回オブジェクト指向プログラミング	

II-14-5. C++言語によるオブジェクト指向プログラミング

C++によるオブジェクト指向プログラミングの実現方法について解説する。C++におけるクラスの定義方法、コンストラクタ、デストラクタ、クラスの継承、オーバーライド、オーバーロード、多重継承などについて説明する。

【学習の要点】

* オブジェクトは状態を保持するためのプロパティと、オブジェクトの動作を定義するためのメソッドで構成され、これらをクラスによって定義する。
* クラスには、初期化を行うコンストラクタ、終了処理を行うためのデストラクタという特殊なメソッドを定義することができる。
* 既存のクラスを元に新しいクラスを作成することをクラスの継承といい、オーバーライドやオーバーロードによって元となるクラスのメソッドを再定義することができる。

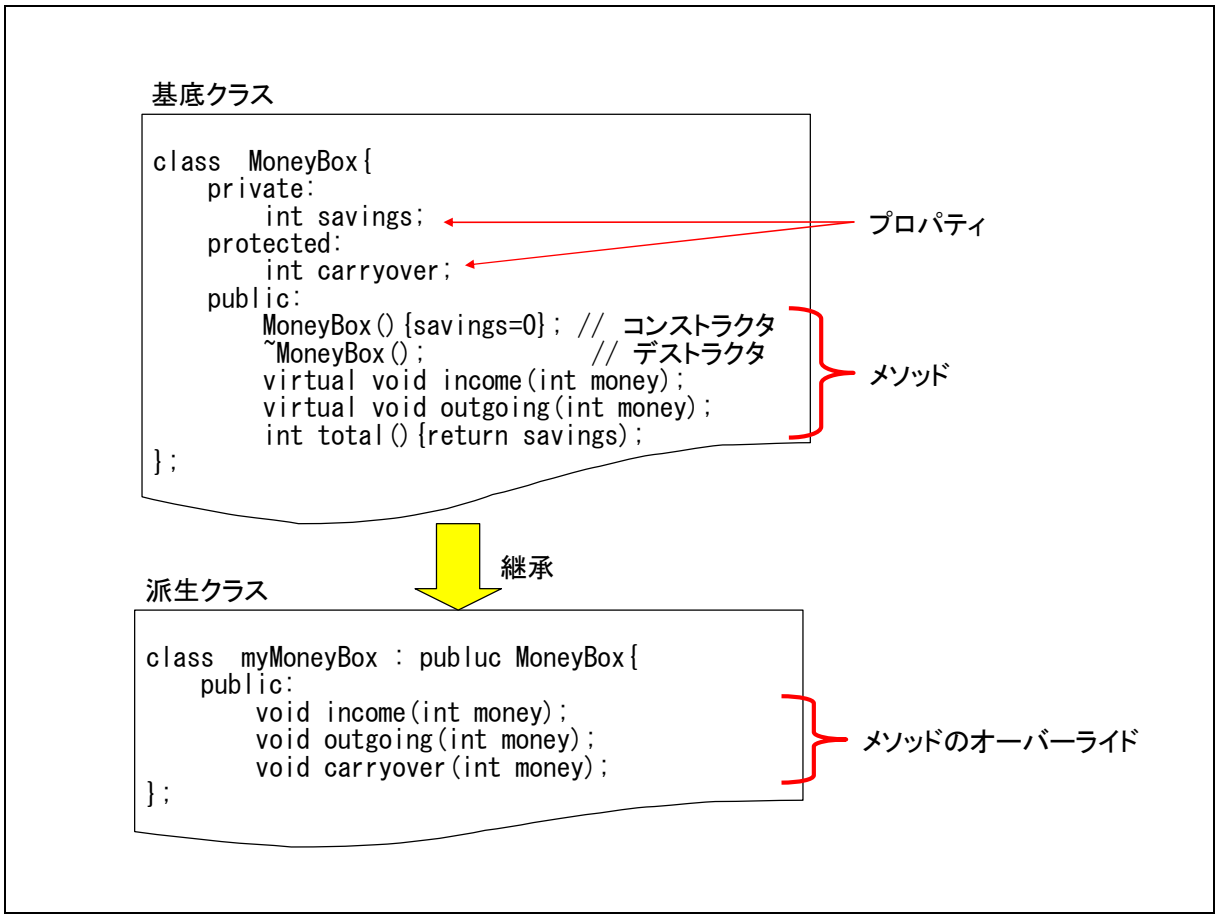

図 II-14-5.　クラスの定義と継承

【解説】

1) クラスの定義

クラスはオブジェクトの状態を保持するためのプロパティ(メンバ変数)と、オブジェクトの動作を定義するためのメソッド(メンバ関数)を定義したものである。
通常は1つのソースファイルに1つのクラスの定義を記述する。
ソースファイルは、クラスのプロパティとメソッドのプロトタイプを記述した宣言部分と各メソッドの本体を記述する関数部分に分けられる。
宣言部分では「class」キーワードにより「class クラス名」のようにクラスの宣言を行う。アクセス識別子(public、private、protected)を指定し、プロパティおよびメソッドのアクセス制限を設定することができる。

- public :クラス外の誰でもアクセス可能
- private :クラス内のみアクセス可能
- protected :クラス内および派生クラス内のみアクセス可能

関数部分ではスコープ解決演算子「:: 」を用いて「クラス名::メソッド名」のように関数の記述を行う。

2) コンストラクタとデストラクタ

クラスからインスタンスを生成する時に呼び出される関数をコンストラクタ、インスタンスが破棄されるときに呼び出される関数をデストラクタと呼ぶ。
コンストラクタの名称はクラスの名称と同じもので定義する。コンストラクタは通常の関数と同様に引数を持つことが可能であるが、戻り値を持たない関数である。
デストラクタの名称はクラスの名称の前にチルダ「~」をつけたもので定義する。デストラクタは引数も戻り値も持たない関数である。
コンストラクタとデストラクタを明示的に定義していない場合、自動的に生成される。

3) クラスの継承

クラスはオブジェクトの属性や操作の雛形を定義したものであり、オブジェクト指向の要素である継承により具体的なオブジェクトを定義する新しいクラスを定義することができる。
継承元となるクラスを「基底クラス」、継承により作成したクラスを「派生クラス」と呼ぶ。
新しいクラスの固有の特性を持たせるため、オーバーライドやオーバーロードによって基底クラスのメソッドを再定義することができる。

* オーバーライドとオーバーロード
  オーバーライドは基底クラスのメソッド名と同一の名称のメソッドを派生クラスで再定義することである。これにより、派生クラスの操作に特化した機能を同一のメソッド名で定義できる。
  オーバーロードは同一クラス内でメソッド名が同一で引数の型、数、並び順が異なるメソッドを複数定義することである。型の異なる変数に対して同じメソッド名で処理を行うことができる。
* 多重継承
  複数のクラスから継承して新しいクラスを定義することを多重継承といい、異なる特性を持つ基底クラスを統合し、複数の特性を持つクラスを作成することができる。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-6. Standard Template Library	
対応する コースウェア	第 13 回 STL（Standard Template Library）	

II-14-6. Standard Template Library

コンテナ、イテレータ、アルゴリズムのライブラリ化といった概念を説明し、C++によるその実装である STL (Standard Template Library)を紹介する。また STL によるプログラミング例を挙げ、STL の具体的な使い方を解説する。

【学習の要点】

* STL(Standard Template Library)は C++言語のテンプレート機能を利用して実装された コンテナ、アルゴリズム、イテレータ(反復子)で構成される標準的なライブラリである。
* コンテナとは他のオブジェクトを格納するための入れ物で、ベクトル、リスト、マップなどがある。
* アルゴリズムとは、アルゴリズムを配列やリストなどのデータ構造に依存しない形で実現に依存しない形で汎用化したものである。
* イテレータとはコンテナの種類に関係なく同じアルゴリズムを適用するために実装するクラスである。

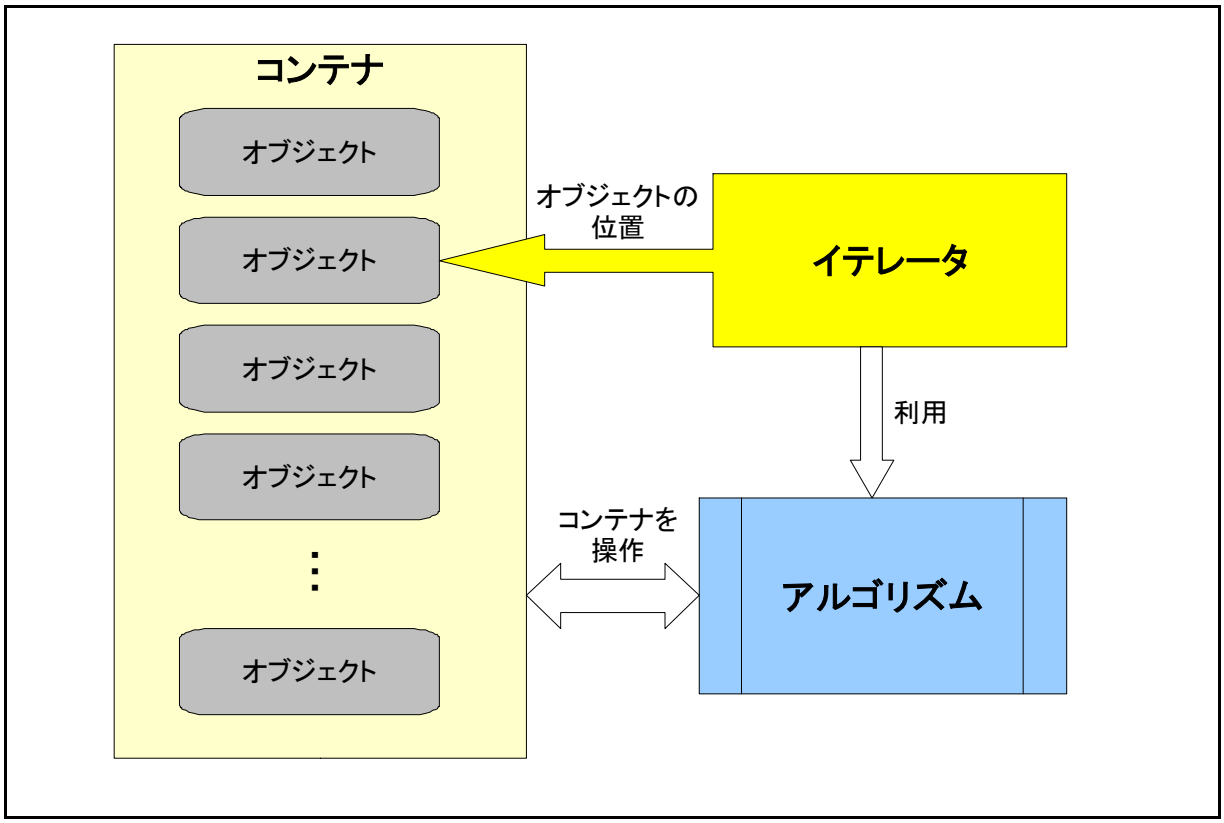

図 II-14-6.　STL の構成

【解説】

1) STL(Standard Template Library)とは

プログラムの作成では同じ処理を記述することが多いため、よく利用される処理をライブラリとしてまとめたものが STL である。STL は C++のテンプレート機能を利用して作成されており、ヘッダファイルのインクルードにより使用できる。

2) コンテナ

STL の中心となるもので、さまざまなデータを格納する。コンテナはデータが順番に格納される順序コンテナと、キーによって自由にアクセスできる連想コンテナの 2 つに分類される。STL の基本的なコンテナには以下のようなものがある。

- vector :ランダムにアクセス可能な順序コンテナ。
- deque :vector と同様のコンテナでデータの挿入や削除の操作を高速に行える。
- list :二重リンクリスト(双方向連結リスト)。ランダムアクセスには対応していない。
- set :オブジェクトのセット。セット内のオブジェクトは順序付けされている。
- multiset :同じオブジェクトを複数格納できるセット
- map :キー値でオブジェクトを検索できるコンテナ。各キーは 1 つだけ値を格納する。
- multimap :各キーに複数の値を格納できるマップ。

3) イレテータ(反復子)

イテレータはコンテナに格納されたデータへのアクセスを提供する。イテレータには、先頭から末尾の方向へ順番にアクセスする前方イテレータ、逆方法でのアクセスが可能な後方イテレータ、双方向でのアクセスが可能な双方向イテレータ、全てのデータにランダムにアクセスが可能なランダムイテレータがある。
使用できるイレテータはコンテナごとに決められている。

4) アルゴリズム

コンテナを操作するための様々なアルゴリズムを実際のデータ構造(配列、リストなど)に依存しないように汎用化したもので、以下のようなアルゴリズムがある。

- find :コンテナ内の要素を検索する。
- count :コンテナ内の要素をカウントする。
- equal :2つのデータが等しいかどうかを評価する。
- for_each :コンテナの各データに対して指定した関数を実行する。
- copy :コンテナをコピーする。
- reverse :順序を持つコンテナのオブジェクトの順序を逆にする。

5) STL 使用上の注意

STL では柔軟性があると同時に、コンテナや関数に多くの型を使用しているため、型を間違えて使用することが多くなる傾向にある。そのため、STL のコンテナを直接使用するのでなく、「typedef」文でコンテナの別名を定義して使用することが望ましい。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-7. GTK+による GUI プログラミング	
対応する コースウェア	第 14 回 GUI アプリケーションの開発	

II-14-7. GTK+による GUI プログラミング

GUI アプリケーションを開発するために用意されたウィジェットライブラリである GTK+を紹介する。GTK+の導入方法、基本的な構造、特徴について述べ、GTK+で使われるシグナルとコールバックの概念を説明する。また簡単なサンプルプログラムにより GTK+の使い方を示す。

【学習の要点】

* GTK+（The GIMP Toolkit）は GIMP の実装のために開発されたウィジェットライブラリで、GNOME など多くのソフトウェアプロジェクトで用いられている。
* GTK+のウィジェットとしては、「ウィンドウ」「ボタン」「メニュー」「コンテナ」など、GUI アプリケーションを実現するためのライブラリが数多く提供されている。

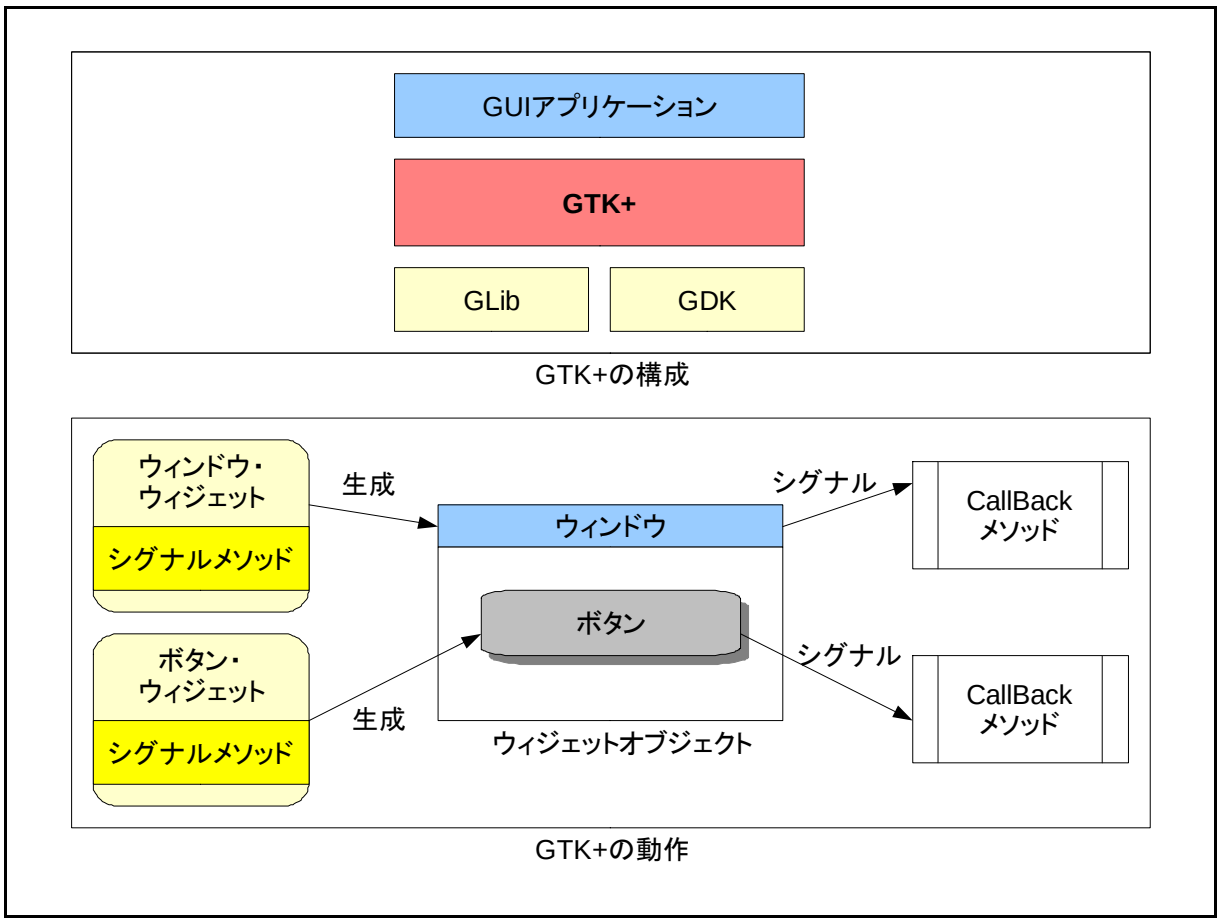

図 II-14-7.　GTK+の構成と動作

【解説】

1) GTK+(The GIMP Toolkit)概要

GTK+は GIMP（GNU Image Manipulation Program）を実装するために開発された、GUI アプリケーションを開発するためのツールキットで、複数のプラットホームに対応している。
C 言語をベースにしているが、オブジェクト指向アーキテクチャが採用されており、C++ではバインディングを用いることで利用することができる。
GTK+は LGPL に準拠したオープンソースライセンスを取っている。
GTK+ は次に示すようなコンポーネント・ライブラリから構成される。

- GLib ： 複数のプットホームで動作するユーティリティライブラリで、多くのデータ型、マクロ、型変換、文字列ユーティリティなどを提供する。
- GDK ： 低レベルのウィンドウ機能のラッパー。

GTK+をインストールするためには、事前に GLib、Pango、ATK、GdkPixbuf、GDK など必要なライブラリをインストールする必要がある。

2) 代表的なウィジェット

ウィジェットは GUI のインターフェイスを構成する要素のことで、クラスの定義により各ウィジェットを定義している。GTK+のウィジェットとしては、「ウィンドウ」「ボタン」「メニュー」「コンテナ」など、GUI アプリケーションを実現するためのライブラリが数多く提供されている。

＊ ウィンドウ・ウィジェットの例
- GtkWindow ：トップレベルのウィンドウの生成を行う。
- GtkDialog ：ポップアップするウィンドウの生成を行う。

＊ ボタン・ウィジェットの例
- GtkButton ：押されたらシグナルを生成するボタンの生成を行う。
- GtkToggleButton ：状態を保持するボタンの生成を行う。

＊ メニュー・ウィジェットの例
- GtkMenu ：ドロップ・ダウン式のメニューを生成する。
- GtkMenuItem ：メニューで使用するアイテムを生成する。

＊ コンテナ・ウィジェットの例
- GtkTable ：ウィジットを（縦、横の）規則正しいパターンの中にパッキングする。
- GtkToolbar ：ボタンとその他のウィジットで構成するバーの生成を行う。

3) GTK+の動作

GTK+はイベント駆動型プログラミング・モデルを採用している。
GTK+ のオブジェクトを生成するとオブジェクト固有のシグナル識別子が割り当てられ、これを処理するためのメソッドへ結び付けられる。イベントが発生した時にシグナルを発行することで通知が行われ、メソッドが呼び出されることにより処理を実現している。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-8. Qt による GUI プログラミング	
対応する コースウェア	第 14 回 GUI アプリケーションの開発	

II-14-8. Qt による GUI プログラミング

GUI アプリケーションを開発するために用意されたもうひとつの著名なウィジェットライブラリである Qt を紹介する。Qt の導入方法、基本的な構造、特徴について述べ、Qt で使われるシグナルとスロットの概念を説明する。また簡単なサンプルプログラムにより Qt の使い方を示す。

【学習の要点】

* Qt は GUI アプリケーションをマルチプラットフォームで開発するためのさまざまなツールを備えたツールキットである。
* Qt のウィジェットライブラリとしては、QButton,QComboBox QDialog QFrame などがある。
* Qt ではオブジェクトが変化した時に発生する「シグナル」と「シグナル」を受け取るための「スロット」によりオブジェクト間の通信を行う。

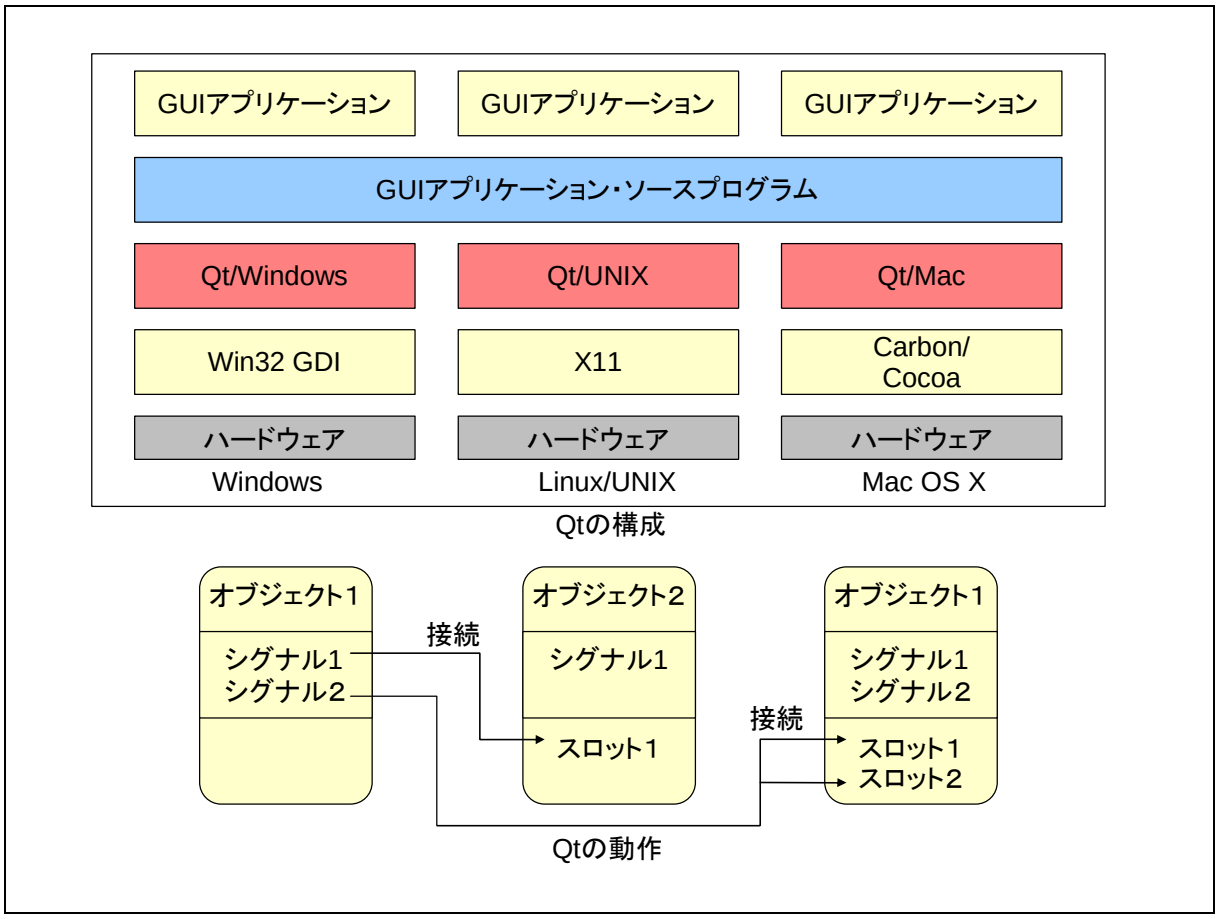

図 II-14-8.　Qt の構成と動作

【解説】

1) Qt 概要

Qt は1つのソースプログラムで Linux や Windows、MacOS といった複数のプラットホームで動作する GUI アプリケーションを開発するための C++用の GUI ツールキットである。
各プラットホーム上で提供される低レベルの描画 API で実装されているため高速に動作するアプリケーションの開発を行うことが可能となる。
GPL に準拠したオープンソース版と商用版のデュアルライセンスを取っている。
開発するプラットホームに Qt のインストールを行い、ソースプログラムにて Qt のヘッダファイルをインクルードし、Qt のライブラリを利用してコンパイルリンクを行うことにより GUI アプリケーションの開発を行う。

2) 代用的なウィジェット

Qt のウィジェットは抽象クラスとして定義されており、これらのクラスを継承することで使用することができる。主なウィジェットとしては以下のようなものがある。

* QButton
  ボタン・ウィジェットの基礎クラスで、ボタンに共通な機能を提供する。
  ユーザの操作に対する反応の仕方とボタンの描画方法はサブクラスで指定する。

* QDialog
  ダイアログウインドウの基礎クラスで、モーダル/モードレス、デフォルトボタン、拡張性、戻り値などの仕組みを提供する。

* QFrame
  フレームを持つことができるウィジェットの基礎クラスで、フレームを描画し、フレームの内容を描画する仮想関数の呼出しを行う。

* QComboBox
  ボタンとポップアップリストを組み合わせたウィジェットのクラスで、選択されているアイテムを表示し、可能な選択肢のリストをポップアップすることができる。

3) Qt の動作

Qt では「シグナル」「スロット」を利用してオブジェクト間の通信を行うことにより GUI の処理を実現している。シグナルとスロットは共にクラスのメソッドとして定義する。クラスの定義にて「Q_OBJECT」を記述することによりこれらを使用するクラスとして認識する。

* シグナル
  シグナルは他のオブジェクトへの通知を行うもので、クラスにてシグナルを発生させるメソッドを定義することにより使用できる。オブジェクトの状態が変化した時にシグナルを発生させるメソッドを呼び出すと、接続されたスロットが通常の関数の呼出と同様に実行される。
  シグナルとなるメソッドは、アクセス識別子に「signal」を指定する。

* スロット
  スロットはシグナルの処理を行うための C++の関数で、接続されたシグナルが発生した時に呼び出される。
  シグナルとなるメソッドは、アクセス識別子に「slot」を指定する。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-9. ライブラリの利用例 1 (DB ライブラリ)	
対応する コースウェア	第 15 回開発ライブラリの使用	

II-14-9. ライブラリの利用例 1 (DB ライブラリ)

そもそもライブラリとは何かを説明し、ライブラリの活用例としてデータベースライブラリの利用方法を紹介する。ライブラリを介してデータベースを操作する方法、データの出し入れを行うためのサンプルプログラムを示し、データベースライブラリの使い方を解説する。

【学習の要点】

* 特定の機能を持ったプログラムを他のプログラムから利用できるように一つのファイルにまとめたものがライブラリである。
* データベースシステム毎に提供されているデータベースライブラリの機能を用いることで容易にデータベースを利用するプログラムが作成できる。

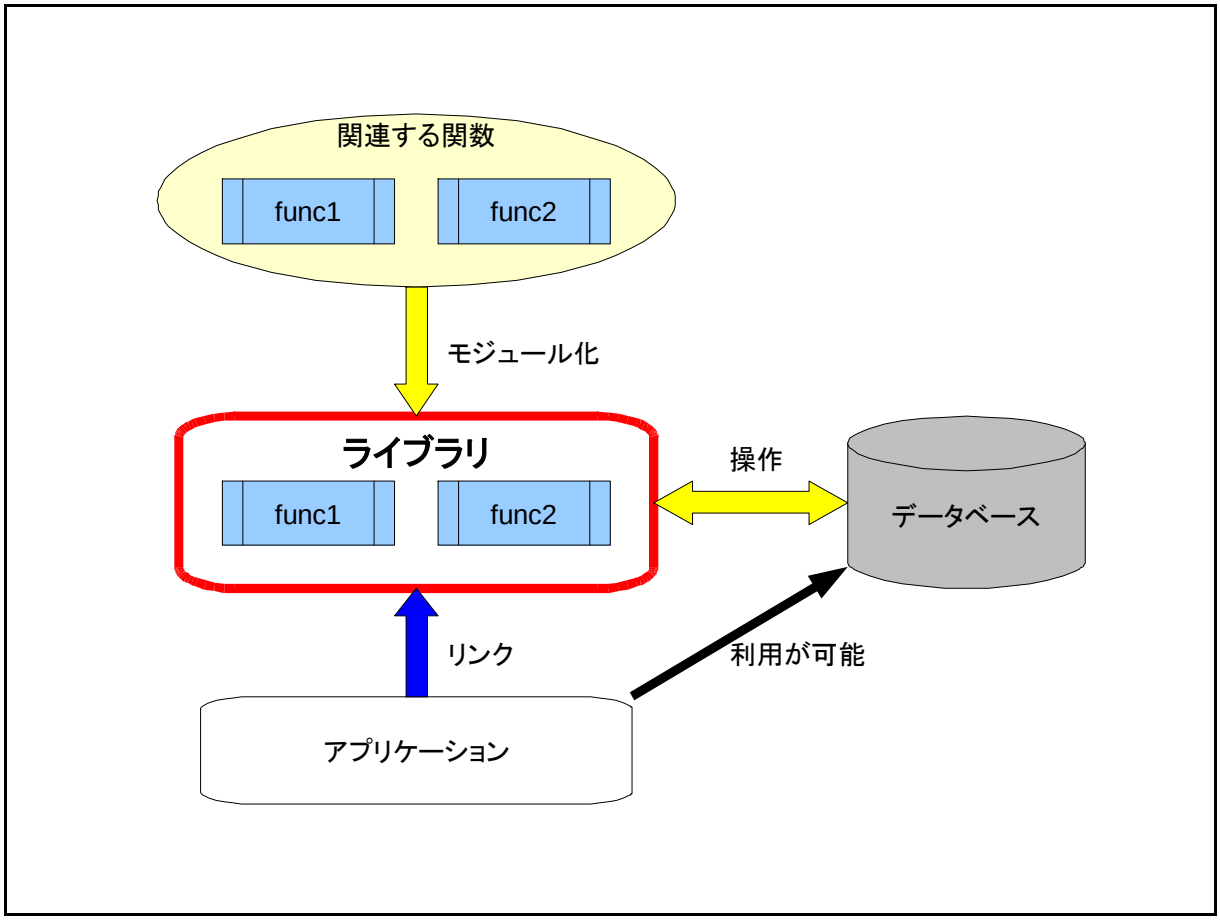

図 II-14-9.　ライブラリの作成と利用

【解説】

1) ライブラリとは

C++を使用する利点として、プロパティ(メンバ変数)とメソッド(メンバ関数)で記述された処理を再利用可能なコンポーネントとしてまとめられる点がある。作成するプログラムを他から流用することができれば、プログラムを効率よく作成することができる。
これを実現するため、特定の機能を持った汎用的な関数をモジュールという形式にまとめ、このモジュールを再利用可能な状態にまとめたものがライブラリである。
プログラムにてライブラリを利用する形態として、実行ファイルに含まれず、プログラムが実行されるときに読み込まれる「動的リンク」と実行ファイルに含められる「静的リンク」がある。なお、ライブラリを利用する場合には、ライブラリのライセンスにも配慮しなくてはならない(詳細は I-2-5 参照)。

2) データベースライブラリ

多くのデータを扱うプログラムでは効率よくデータの管理を行うため、データベースを用いる。プログラムからデータベースを利用する方法としては、データベースをプログラムの一部として組み込む方法や、MySQL や PostgreSQL などデータベースアプリケーションを利用する方法がある。
いずれの方法でも、データベースを利用するためのライブラリが提供されており、これを使用することで容易にプログラムを作成することができる。プログラムに組み込むためのライブラリとしては BerkeleyDB ライブラリや SQLite ライブラリ、データベースアプリケーションを利用するためのライブラリとしては libpq++などがある。

＊ BerkeleyDB ライブラリ
BerkeleyDB は C 言語で実装された、プログラム組み込み型のデータベースで、データはキーと対応する値のペアという単純な形式で格納するため、高速で動作すると共にプログラムのサイズが小さいことが特徴である。
データベースへのアクセスは全て関数の呼出しにより行い、C 言語または C++から利用することができる。

＊ SQLite ライブラリ
SQLite はサーバを必要としないプログラム組み込み型のデータベースで、ユーザの概念がなく、単一のファイルで構成され、保存するデータの形式はシステムに依存しないため、可搬性が高いことが特徴である。
SQLite ライブラリの呼出しにより SQL を実行してデータの参照や保存を行うため、データベースを意識することなく、ファイルの入出力のように簡単に利用することができる。

＊ libpq++ライブラリ
libpq++ は PostgreSQL のための C++ API で、コネクションクラスとデータベースクラスで構成される。コネクションクラスは実際のデータベースサーバとの接続を確立するために使用し、データベースへアクセスする全てのクラスがこのクラスを継承する。データベースクラスはデータベースへの接続と問合せを行うために使用する。
データベースへの問合せは、データベースクラスにて SQL 文をデータベースサーバへ送信することにより行う。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	14 C、C++に関する知識 II	応用
習得ポイント	II-14-10. ライブラリの利用例 2 (オプション解析ライブラリ)	
対応する コースウェア	第 15 回開発ライブラリの使用	

II-14-10. ライブラリの利用例 2 (オプション解析ライブラリ)

もうひとつのライブラリ利用例として、コマンドラインオプションを解析するライブラリの使い方を紹介する。コマンドラインオプションとは何か、どのようなスタイルがあるかを説明し、コマンドラインオプション解析ライブラリを用いてプログラムからコマンドラインオプションを解析するやり方を説明する。

【学習の要点】

* プログラムの実行時に必要なパラメータを自由に設定できるようにするためには、コマンドラインのオプションとして設定できるようにすることが有効である。
* popt、getopt などのコマンドラインのオプションを解析するライブラリを用いることで統一性のあるオプション設定処理が実現できる。

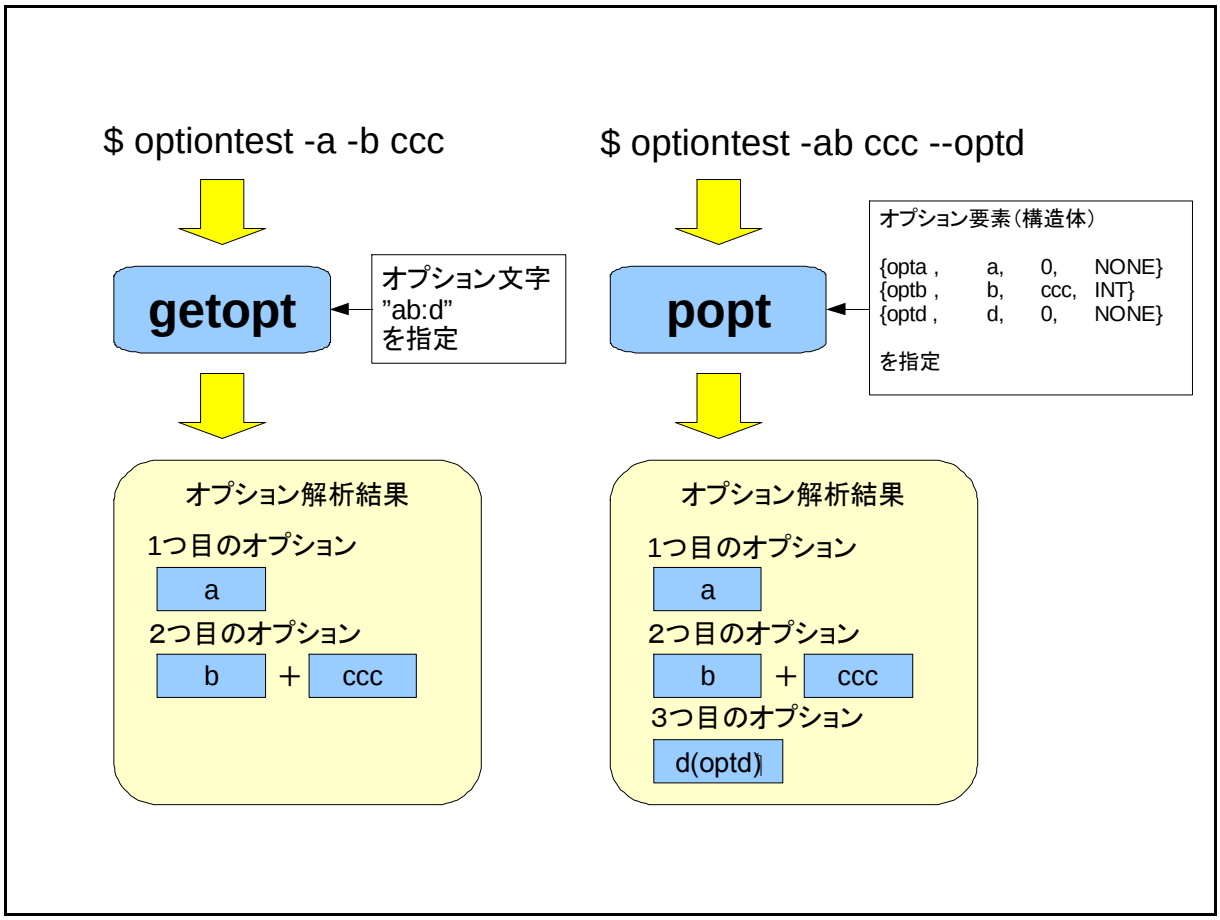

図 II-14-10.　getopt、popt によるオプション解析

【解説】

プログラムの実行時コマンドラインのオプションにてパラメータを受け取るようにする場合、main 関数の引数により設定されたオプションを受け取ることが可能である。
作成するプログラムでオプションを解析する処理を記述した場合、複数のオプションを扱えるようにするにはオプション 1 つずつに処理を作成していく必要がある。
オプションを解析するライブラリとして getopt や popt といったライブラリが提供されており、これを利用することにより、容易にオプションの解析を行うことが可能となる。

1) getopt

コマンドラインのオプションとして設定された文字列は、「-」や「--」の後に続く「オプション要素」とこれに続いて設定される「引数」の 2 つに大別して認識する。
getopt はオプションを解析した結果を返す getopt 関数と、引数を返す optarg グローバル変数によって構成され、getopt 関数は以下のように引数を指定して使用し、繰り返し呼び出されるごとに次のオプションを返す。

```
opt = getopt(argc, argv, オプション文字);
```

argc,argv ： main 関数の引数によって取得したオプション数とオプション文字列
オプション文字：オプション要素として指定できる文字の文字列
オプション要素で引数を持つものは、文字の後に「:」をつける。
-a -b ccc のように指定する場合、”ab:” となる。

getopt 関数にて argv にオプション文字として指定した文字が見つかると、opt にその文字を返す。見つかったオプションに引数が必要な場合、optarg グローバル変数へ引数の文字列を設定する。argv に指定した文字以外のものが見つかると、「?」文字を返す。argv の最後まで解析が終わった場合、-1を返す。

2) popt

getopt と同様にオプションの解析を行うライブラリであるが、以下の点が優れている。

- グローバル変数を使わない
- --help などの長いオプションを扱える
- --help と --usage のメッセージを自動生成する

popt ではオプション要素として、「長いオプション」「オプション文字」「引数の有無と型」「オプションの説明」などを構造体として定義する。
poptGetContext 関数へ main 関数の引数によって取得したオプション数とオプション文字列およびオプション要素として定義した構造体を引数として指定すると、解析結果を返す。

```
popContext = poptGetContext(NULL, argc, argv, オプション構造体,0);
```

poptGetNextOpt 関数へ解析結果を引数として指定し、繰り返し呼び出すことにより、指定されたオプションを取り出すことができる。解析結果の最後に到達した場合、-1 を返す。

```
rc = poptGetOpt(popContext);
```

また引数については、poptGetArg 関数または poptPrrkArg 関数にて取得することができる。

```
rcChar = poptGetArg(popContext);
```