1.1 サンプリング

図 1.1（a）に示すように、本来、音はすべての時刻で値をとる「アナログ信号」の物理現象ですが、コンピュータはこうしたアナログ信号をそのままの状態では取り扱うことができません。図 1.1（b）に示すように、コンピュータで音を取り扱うには、アナログ信号の音を離散的な時刻でしか値をとらない「ディジタル信号」の音データに変換する必要があります。こうした処理を「サンプリング」と呼びます。

サンプリングは、アナログ信号をディジタル信号に変換する処理になっているため、「A-D（Analog-to-Digital）変換」とも呼ばれます。一方、コンピュータに記録された音データをスピーカーから再生するには、ディジタル信号の音データをアナログ信号の音に戻す必要があります。このように、ディジタル信号をアナログ信号に変換する処理を「D-A（Digital-to-Analog）変換」と呼びます。

1.2 標本化

サンプリングは、「標本化周期」と呼ばれる時間間隔でアナログ信号を読

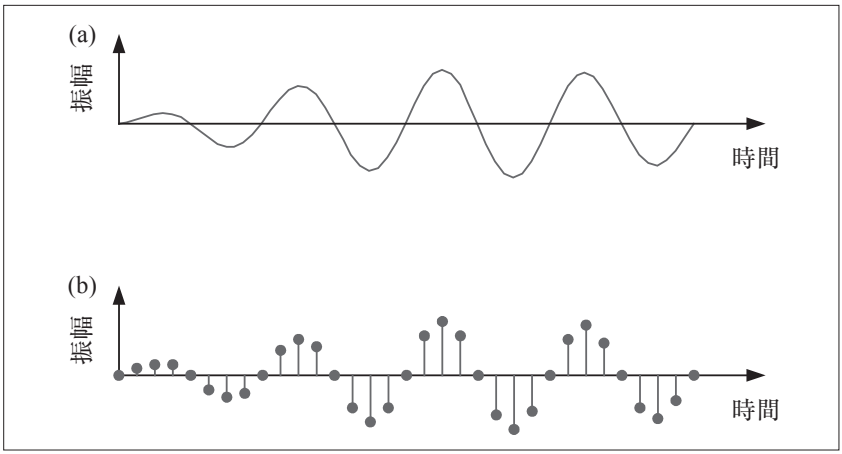

図 1.1　サンプリング：(a) アナログ信号、(b) ディジタル信号

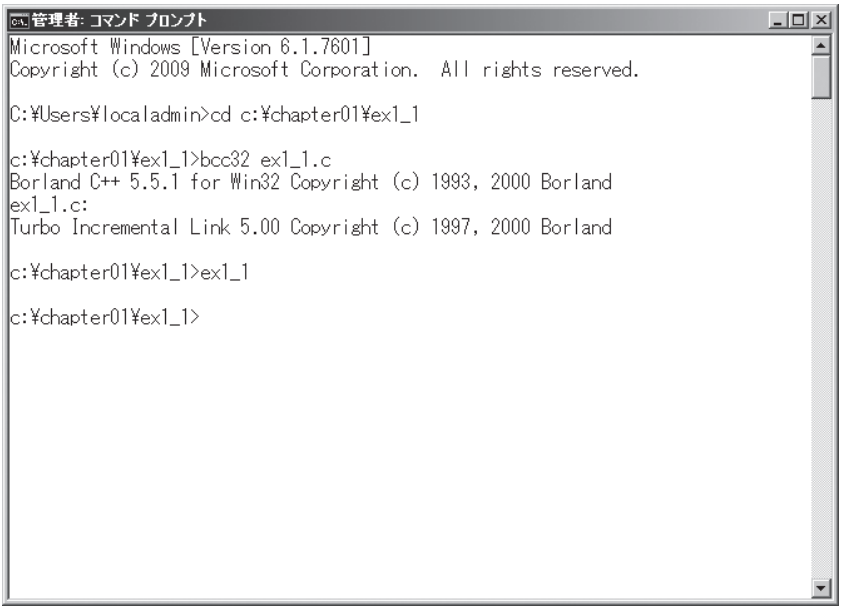

図 1.13　Borland C++ Compiler 5.5 によるプログラムのコンパイルと実行

c をコンパイルするため、コマンドプロンプトに次のように入力してください。

```
> cd c:¥chapter01¥ex1_1 ↵
> bcc32 ex1_1.c ↵
```

図 1.13 に示すようなメッセージが表示されれば、ex1_1.c は問題なくコンパイルされたことになり、ex1_1.exe という実行ファイルが生成されます。ex1_1.exe を実行するため、コマンドプロンプトに次のように入力してください。

```
> ex1_1 ↵
```

ex1_1.exe を実行すると、ex1_1 フォルダに b.wav が生成されます。この実行ファイルは、a.wav の音データを読み取り、その内容をコピーして b.wav

に書き出すものになっています。

実際にこれらの WAVE ファイルを聞いてみましょう。「スタート」→「すべてのプログラム」→「Windows Media Player」を順に選択し、Windows Media Player を起動してください。次に、Windows Media Player のメニューから「ファイル」→「開く」を選択し、ex1_1 フォルダの a.wav と b.wav をそれぞれ聞き比べてみましょう。どちらもまったく同じ音になっていることがおわかりいただけるでしょうか。

プログラムの動作が確認できたところで、改めて ex1_1.c の内容について調べてみることにしましょう。**リスト 1.1** に ex1_1.c を示します。

ex1_1.c は、MONO_PCM 型の構造体によってモノラルの音データを取り

リスト 1.1　ex1_1.c

```
#include <stdio.h>
#include <stdlib.h>
#include "wave.h"

int main(void)
{
    MONO_PCM pcm0, pcm1;
    int n;

    wave_read_16bit_mono(&pcm0, "a.wav"); /* 音データの入力 */

    pcm1.fs = pcm0.fs; /* 標本化周波数 */
    pcm1.bits = pcm0.bits; /* 量子化精度 */
    pcm1.length = pcm0.length; /* 音データの長さ */
    pcm1.s = calloc(pcm1.length, sizeof(double)); /* 音データ */

    for (n = 0; n < pcm1.length; n++)
    {
        pcm1.s[n] = pcm0.s[n]; /* 音データのコピー */
    }

    wave_write_16bit_mono(&pcm1, "b.wav"); /* 音データの出力 */

    free(pcm0.s);
    free(pcm1.s);

    return 0;
}
```