第 6 章　使用 C++的财务计算

尽管那些庞大而成熟的应用程序，如编译器、Internet 浏览器、文字处理器、数据库以及财务软件包等，占据了计算的主流，但是仍然存在一类普遍而简短的程序。这些程序执行不同的财务计算，如计算贷款的定期偿还、投资的预期价值或者贷款余额。这些计算都不复杂，也不需要很多的代码，然而它们产生的信息却非常有用。

由于 C++的特长在于建立高效的系统软件，因此对于财务应用程序，它经常被忽略，然而这是一个失误。C++在这个领域内有卓越的性能，它支持一整套数学函数和高效的浮点数运算。另外，由于它可以生成特别快速的执行代码，因此 C++非常适合于执行复杂的财务分析和建模的程序。

为了说明 C++能够轻松处理财务计算，本章开发了许多小程序来执行如下所示的财务计算：

(1) 贷款的定期偿还
(2) 投资的预期价值
(3) 为了获得预期的价值所需的初始投资
(4) 为了获得预期的养老金所需的投资
(5) 投资所能得到的养老金的最大值
(6) 贷款余额

这些程序可以直接使用，也可以将其修改以适应您的需要。坦白地说，尽管它们是这本书的最简单程序，但是它们可能也是您使用次数最多的程序。

6.1　计算贷款的定期偿还

最常用的财务计算可能就是计算贷款的定期偿还，如汽车或者住房贷款。贷款的偿还金额用下面的公式来计算：

$$\text{Payment} = (\text{intRate} * (\text{principal} / \text{payPerYear})) / (1 - ((\text{intRate} / \text{payPerYear}) + 1)^{-(\text{payPerYear} * \text{numYears})})$$

其中 intRate 为利率，principal 为本金，payPerYear 为每年偿还的数额，numYears 为贷款年限。

在下面的程序中，函数 regpay()使用前面公式计算了贷款的偿还金额。本金、利率、贷款年限以及每年的偿还金额传递给这个函数。这个函数返回要偿还的金额。

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <locale>
```

```
using namespace std;

// Compute the regular payments on a loan.
double regpay(double principal, double intRate,
              int numYears, int payPerYear) {
  double numer;
  double denom;
  double b, e;

  intRate /= 100.0; // convert percentage to fraction

  numer = intRate * principal / payPerYear;

  e = -(payPerYear * numYears);
  b = (intRate / payPerYear) + 1.0;

  denom = 1.0 - pow(b, e);

  return numer / denom;
}

int main() {
  double p, r;
  int y, ppy;

  // Set locale to English. Adjust as necessary
  // for your language and/or region.
  cout.imbue(locale("english"));

  cout << "Enter principal: ";
  cin >> p;

  cout << "Enter interest rate (as a percentage): ";
  cin >> r;

  cout << "Enter number years: ";
  cin >> y;

  cout << "Enter number of payments per year: ";
  cin >> ppy;

  cout << "\nPayment: " << fixed << setprecision(2)
       << regpay(p, r, y, ppy) << endl;

  return 0;
}
```

为了计算贷款的偿还金额，只需要按提示输入所需的信息。下面是一个运行的样本：

```
Enter principal: 1000
Enter interest rate (as a percentage): 9
Enter number years: 5
```

```
Enter number of payments per year: 12

Payment: 20.76
```

在 main()函数中，有两点值得注意。首先，将与 cout 相关的 locale 设置为英语。这是通过调用成员函数 imbue()并向 locale 对象传递 English 来完成的。这意味着货币值将使用英语的习惯来显示，也就是以千为单位使用逗号来分割，使用句点作为小数点。其次，在显示贷款的偿还额之前，改变了数值的显示格式，将精度固定地设置为 2。这使得输出的小数部分有两位，并且正确地取整。如果需要的话，小数部分也可以用 0 来补齐。所有的财务计算程序都使用了相同的方法。为了改变格式以适应不同的语言或者地区，只需要修改传递给 imbue()的 locale 对象的语言/地区。

6.2 计算投资的预期价值

另一个常用的财务计算是计算给定初始投资的预期价值、回报率、每年的复利计算数额以及投资的年限。例如，如果本金为 98,000 美元，平均每年返回 6%，您可能想要知道 12 年后您的退休金的数额。这里给出的程序将提供答案。

为了计算这个未来值，使用下面的公式：

$$\text{Future Value} = \text{principal} * ((\text{rateOfRet} / \text{compPerYear}) + 1)^{\text{compPerYear} * \text{numYears}}$$

在此，rateOfRet 是返回率，principal 包含了原始的投资值，compPerYear 指定了每年复利计算期的数目，numYear 指定了投资的年限。如果您对 rateOfRet 使用按年返回率，那么复利计算期的数目为 1。

在下面的程序中，函数 futval()使用前面的公式来计算投资的未来价值。本金、返回率、投资的年限以及每年复利计算的数量传递给这个函数。函数返回预期价值。

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <locale>

using namespace std;

// Compute the future value of an investment.
double futval(double principal, double rateOfRet,
              int numYears, int compPerYear) {
  double b, e;

  rateOfRet /= 100.0; // convert percentage to fraction

  b = (1 + rateOfRet/compPerYear);
  e = compPerYear * numYears;

  return principal * pow(b, e);
}
```

```
int main() {
  double p, r;
  int y, cpy;

  // Set locale to English. Adjust as necessary
  // for your language and/or region.
  cout.imbue(locale("english"));

  cout << "Enter principal: ";
  cin >> p;

  cout << "Enter rate of return (as a percentage): ";
  cin >> r;

  cout << "Enter number years: ";
  cin >> y;

  cout << "Enter number of compoundings per year: ";
  cin >> cpy;

  cout << "\nFuture value: " << fixed << setprecision(2)
       << futval(p, r, y, cpy) << endl;

  return 0;
}
```

下面是运行示例：

```
Enter principal: 10000
Enter rate of return (as a percentage): 6
Enter number years: 5
Enter number of compoundings per year: 12

Future value: 13,488.50
```

6.3 计算为了获得预期的价值所需的原始投资

有时候您想要知道为了获得预期的价值所需的原始投资的数额。例如，如果您想要为孩子的大学教育储蓄，并且您知道 5 年内需要 75,000 美元，当利率为 7%的时候，您需要投资多少钱来达到这个目标？下面开发的这个程序可以回答这个问题。

计算原始投资的公式如下：

$$\text{Initial Investment} = \text{targetValue} / (((\text{rateOfRet} / \text{compPerYear}) + 1)^{\text{compPerYear} * \text{numYears}})$$

在此 rateOfRet 指定了返回率，targetValue 包含了需要的预期价值，compPerYear 指定了每年复利计算期的数目，numYears 指定了投资的年限。如果您对 rateOfRet 使用年返回率，则复利计算期的数目为 1。

在下面的程序中，函数 initval()使用了前面的公式来计算为了获得预期价值所需的原始投资。目标价值、返回率、投资的年限以及每年复利计算的数目都传递给这个函数。函数返回为了获得预期价值所需的原始投资。

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <locale>

using namespace std;

// Compute the initial investment necessary for
// a specified future value.
double initval(double targetValue, double rateOfRet,
               int numYears, int compPerYear) {
  double b, e;
  rateOfRet /= 100.0; // convert percentage to fraction

  b = (1 + rateOfRet/compPerYear);
  e = compPerYear * numYears;

  return targetValue / pow(b, e);
}

int main() {
  double p, r;
  int y, cpy;

  // Set locale to English. Adjust as necessary
  // for your language and/or region.
  cout.imbue(locale("english"));

  cout << "Enter desired future value: ";
  cin >> p;

  cout << "Enter rate of return: ";
  cin >> r;

  cout << "Enter number years: ";
  cin >> y;

  cout << "Enter number of compoundings per year: ";
  cin >> cpy;

  cout << "\nInitial investment required: "
       << fixed << setprecision(2)
       << initval(p, r, y, cpy) << endl;

  return 0;
}
```

下面是运行的样本：

```
Enter desired future value: 75000
Enter rate of return (as a percentage): 7
Enter number years: 5
Enter number of compoundings per year: 4

Initial investment required: 53,011.84
```

6.4 为了获得预期的养老金所需的原始投资

另一个常用的财务计算是计算为了能够定期获取预期数量的养老金必须初期投资的数量。例如，您在退休时可能每个月需要 5,000 美元的退休金，并且需要 20 年。问题是为了保证这些养老金，您需要投资多少钱呢？答案可以用下面的公式计算出来：

$$\text{Initial Investment} = ((\text{regWD} * \text{wdPerYear}) / \text{rateOfRet}) * (1 - (1 / (\text{rateOfRet} / \text{wdPerYear} + 1)^{\text{wdPerYear} * \text{numYears}}))$$

在此，rateOfRet 指定了返回率，regWD 包含了需要的定期提取的数量，wdPerYear 指定了每年提取的次数，numYears 指定了养老金的年限。

在下面的程序中，函数 annuity()计算了为了获得预期的养老金所需的初期投资。所要提取的数量、返回率、养老金的年限以及每年复利计算数目传递给这个函数。函数返回为了获得养老金所需的最少投资。

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <locale>

using namespace std;

// Compute the initial investment necessary for
// a desired annuity. In other words, it finds
// the initial amount needed to allow the regular
// withdrawals of a desired amount over a period
// of time.
double annuity(double regWD, double rateOfRet,
               int numYears, int numPerYear) {

  double b, e;
  double t1, t2;

  rateOfRet /= 100.0; // convert percentage to fraction

  t1 = (regWD * numPerYear) / rateOfRet;

  b = (1 + rateOfRet/numPerYear);
  e = numPerYear * numYears;
```

```
  t2 = 1 - (1 / pow(b, e));

  return t1 * t2;
}

int main() {
  double wd, r;
  int y, wpy;

  // Set locale to English. Adjust as necessary
  // for your language and/or region.
  cout.imbue(locale("english"));

  cout << "Enter desired withdrawal: ";
  cin >> wd;

  cout << "Enter rate of return (as a percentage): ";
  cin >> r;

  cout << "Enter number years: ";
  cin >> y;

  cout << "Enter number of withdrawals per year: ";
  cin >> wpy;

  cout << "\nInitial investment required: "
       << fixed << setprecision(2)
       << annuity(wd, r, y, wpy) << endl;

  return 0;
}
```

下面是运行样本：

```
Enter desired withdrawal: 5000
Enter rate of return (as a percentage): 6
Enter number years: 20
Enter number of withdrawals per year: 12

Initial investment required: 697,903.86
```

6.5 计算给定投资所能得到的养老金的最大值

另一个养老金计算问题是计算给定的投资在一段时期之后可以得到的养老金的最大值（定期提取）。例如，如果您有 500,000 美元的退休账户，假定返回率为 6%，20 年内您每个月可以提取多少美元呢？计算最大提取金额的公式如下所示：

$$\text{Maximum Withdrawal} = \text{principal} * \left(\left(\left(\text{rateOfRet} / \text{wdPerYear}\right) / \left(-1 + \left(\left(\text{rateOfRet} / \text{wdPerYear}\right) + 1\right)^{\text{wdPerYear} * \text{numYears}}\right)\right) + \left(\text{rateOfRet} / \text{wdPerYear}\right)\right)$$

其中 rateOfRet 指定了返回率，principal 包含了原始投资值，wdPerYear 指定了每年提取的次数，numYears 指定了退休金的年限。

下面所示的程序中，函数 maxwd()计算了在给定返回率和年限的时候，可以定期提取的最大值。将本金、返回率、退休金年限以及每年复利计算的数目传递给这个函数。函数返回最大的养老金数额。

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <locale>

using namespace std;

// Compute the maximum annuity that can
// be withdrawn from an investment over
// a period of time.
double maxwd(double principal, double rateOfRet,
             int numYears, int numPerYear) {

  double b, e;
  double t1, t2;

  rateOfRet /= 100.0; // convert percentage to fraction

  t1 = rateOfRet / numPerYear;

  b = (1 + t1);
  e = numPerYear * numYears;

  t2 = pow(b, e) - 1;

  return principal * (t1/t2 + t1);
}

int main() {
  double p, r;
  int y, wpy;

  // Set locale to English. Adjust as necessary
  // for your language and/or region.
  cout.imbue(locale("english"));

  cout << "Enter principal: ";
  cin >> p;

  cout << "Enter rate of return (as a percentage): ";
```

```
  cin >> r;

  cout << "Enter number years: ";
  cin >> y;

  cout << "Enter number of withdrawals per year: ";
  cin >> wpy;

  cout << "\nMaximum withdrawal: " << fixed << setprecision(2)
       << maxwd(p, r, y, wpy) << endl;

  return 0;
}
```

下面是运行样本：

```
Enter principal: 500000
Enter rate of return (as a percentage): 6
Enter number years: 20
Enter number of withdrawals per year: 12

Maximum withdrawal: 3,582.16
```

6.6 计算贷款余额

您经常想知道贷款的余额。如果您知道本金、利率、贷款期限以及还款的数量，这很容易计算。为了计算贷款余额，您必须对还款数量求和，减去每次还款所分配的利息，然后从本金中减去这个结果。

这里所给出的 **balance()**函数实现贷款计算的余额。将本金、利率、还款数量、每年偿还的次数以及已经偿还的次数传递给这个函数。函数返回贷款余额。

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <locale>

using namespace std;

// Find the remaining balance on a loan.
double balance(double principal, double intRate,
               double payment, int payPerYear,
               int numPayments) {

  double bal = principal;
  double rate = intRate / payPerYear;

  rate /= 100.0; // convert percentage to fraction

  for(int i = 0; i < numPayments; i++)
    bal -= payment - (bal * rate);
```

```
  return bal;
}

int main() {
  double p, r, pmt;
  int ppy, npmt;

  // Set locale to English. Adjust as necessary
  // for your language and/or region.
  cout.imbue(locale("english"));

  cout << "Enter original principal: ";
  cin >> p;

  cout << "Enter interest rate (as a percentage): ";
  cin >> r;

  cout << "Enter payment: ";
  cin >> pmt;

  cout << "Enter number of payments per year: ";
  cin >> ppy;

  cout << "Enter number of payments made: ";
  cin >> npmt;

  cout << "Remaining balance: " << fixed << setprecision(2)
       << balance(p, r, pmt, ppy, npmt) << endl;

  return 0;
}
```

下面是运行样本：

```
Enter original principal: 10000
Enter interest rate (as a percentage): 9
Enter payment: 207.58
Enter number of payments per year: 12
Enter number of payments made: 30

Remaining balance: 5,558.19
```

6.7 尝试完成以下任务

您可能还会发现许多有用的财务计算。例如，计算投资的返回率或者计算为了达到某个未来价值所需的定期存款都是有用的程序。您还可打印一个贷款分期偿还的图表。

您可能会尝试建立一个比较大的应用程序来包含本章提供的所有计算，使得用户从菜单中选择需要的计算。