```cpp
//进制转换器.cpp

#include <iostream>
#include <string>
#include <stack>

using namespace std;

char HexNumToCh(const int& num)
{
    switch (num)
    {
        case 10:
            return 'A';
            break;
        case 11:
            return 'B';
            break;
        case 12:
            return 'C';
            break;
        case 13:
            return 'D';
            break;
        case 14:
            return 'E';
            break;
        default:
            return 'F';
            break;
    }
}

double HexChToNum(char hexCh)
{
    switch (hexCh)
    {
        case 'A':
            return 10;
            break;
        case 'B':
            return 11;
            break;
        case 'C':
            return 12;
            break;
        case 'D':
            return 13;
            break;
        case 'E':
            return 14;
```

```
            break;
        default:
            return 15;
            break;
    }
}

void toNegative(char* dpnArr, const size_t& arrLen)
{
    for (size_t i = 0; i < arrLen; i++)
    {
        if (dpnArr[i] == '1')
        {
            dpnArr[i] = '0';
        }
        else if (dpnArr[i] == '0')
        {
            dpnArr[i] = '1';
        }
        else
        {
            continue;
        }
    }
    size_t index = arrLen - 1;
    while (dpnArr[index] == '1') {
        dpnArr[index] = '0';
        index--;
    }
    dpnArr[index] = '1';
    return;
}

char* DecToDPN(string decimalStr, size_t& arrLen, const unsigned int&
 convert)
{
    double decimal = stod(decimalStr);
    bool sign = decimal >= 0 ? true : false;
    if (!sign)
    {
        decimal *= -1.0;
    }
    bool isDouble = (decimal - (int)decimal == 0 ? false : true);
    stack<char> place;
    int integer = decimal;
    while (true)
    {
        if (integer / convert == 0)
        {
            if (integer % convert < 10)
            {
```

```
            place.push((integer % convert) + '0');
        }
        else
        {
            place.push(HexNumToCh(integer % convert));
        }
        break;
    }
    if (integer % convert < 10)
    {
        place.push((integer % convert) + '0');
    }
    else
    {
        place.push(HexNumToCh(integer % convert));
    }
    integer /= convert;
}
size_t intLen = place.size();
size_t newIntLen = intLen;
while (newIntLen % 8 != 0)
{
    newIntLen++;
}
char* dpnArr = new char[newIntLen];
for (size_t i = 0; i < newIntLen; i++)
{
    dpnArr[i] = '0';
}
size_t index = newIntLen - 1;
stack<char> nPlace;
while (!place.empty())
{
    nPlace.push(place.top());
    place.pop();
}
while (!nPlace.empty())
{
    dpnArr[index] = nPlace.top();
    nPlace.pop();
    index--;
}
if (!sign)
{
    toNegative(dpnArr, newIntLen);
    char* ngtDpnArr = new char[newIntLen + 1];
    ngtDpnArr[0] = '1';
    for (size_t i = 0; i < newIntLen; i++)
    {
        ngtDpnArr[i + 1] = dpnArr[i];
    }
```

```
    delete [] dpnArr;
    dpnArr = ngtDpnArr;
}
if (isDouble)
{
    place.push('.');
    double point = decimal - (int)decimal;
    size_t loopTimes = 0;
    while (loopTimes < 10)
    {
        if (point * convert - (int)point == 0)
        {
            break;
        }
        if ((int)(point * convert) < 10)
        {
            place.push(((int)(point * convert)) + '0');
        }
        else
        {
            place.push(HexNumToCh((int)(point * convert)));
        }
        point = point * convert - (int)(point * convert);
        loopTimes++;
    }
    if (loopTimes >= 9)
    {
        place.push('.');
        place.push('.');
        place.push('.');
    }
    arrLen = newIntLen + place.size();
    char *nDpnArr = new char[arrLen];
    index = newIntLen;
    for (size_t i = 0; i < newIntLen; i++)
    {
        nDpnArr[i] = dpnArr[i];
    }
    delete [] dpnArr;
    dpnArr = nullptr;
    while (!place.empty())
    {
        nPlace.push(place.top());
        place.pop();
    }
    while (!nPlace.empty())
    {
        nDpnArr[index] = nPlace.top();
        nPlace.pop();
        index++;
    }
```

```
        return nDpnArr;
    }
    arrLen = newIntLen;
    if (!sign)
    {
        arrLen++;
    }
    return dpnArr;
}

double DpnToDEC(const string& dpnStr, const int& arrLen, unsigned int
 rPN)
{
    unsigned int oriRPN = rPN;
    int intLen = arrLen;
    bool sign = stod(dpnStr) >= 0 ? true : false;
    int startIndex = 0;
    if (!sign)
    {
        startIndex = 1;
    }
    bool isDouble = false;
    for (size_t i = startIndex; i < arrLen; i++)
    {
        if (dpnStr[i] == '.')
        {
            isDouble = true;
            intLen = (int)i;
            break;
        }
    }
    double decimal = 0;
    for (int i = intLen - 1; i >= startIndex; i--)
    {
        if (i == intLen - 1)
        {
            if (dpnStr[i] >= 'A' && dpnStr[i] <= 'F')
            {
                decimal = HexChToNum((int)dpnStr[i]);
            }
            else
            {
                decimal = ((int)dpnStr[i]) - 48;
            }
            continue;
        }
        if (i == intLen - 2)
        {
            if (dpnStr[i] >= 'A' && dpnStr[i] <= 'F')
            {
                decimal += (HexChToNum((int)dpnStr[i])) * rPN;
```

```
        }
        else
        {
            decimal += (((int)dpnStr[i]) - 48) * rPN;
        }
        continue;
    }
    rPN *= oriRPN;
    if (dpnStr[i] >= 'A' && dpnStr[i] <= 'F')
    {
        decimal += (HexChToNum((int)dpnStr[i])) * rPN;
    }
    else
    {
        decimal += ((int)dpnStr[i] - 48) * rPN;
    }
}
rPN = oriRPN;
if (isDouble)
{
    double point = 0;
    for (int i = intLen + 1; i < arrLen; i++)
    {
        if (i == intLen + 1)
        {
            if (dpnStr[i] >= 'A' && dpnStr[i] <= 'F')
            {
                point += HexChToNum(dpnStr[i]) * (1.0 /
                 (double)rPN);
            }
            else
            {
                point += ((double)dpnStr[i] - 48.0) * (1.0 /
                 (double)rPN);
            }
            continue;
        }
        rPN *= oriRPN;
        if (dpnStr[i] >= 'A' && dpnStr[i] <= 'F')
        {
            point += HexChToNum(dpnStr[i]) * (1.0 / (double)rPN);
        }
        else
        {
            point += ((double)dpnStr[i] - 48.0) * (1.0 /
             (double)rPN);
        }
    }
    decimal += point;
}
if (!sign)
```

```
    {
        decimal *= -1;
    }
    return decimal;
}

bool isVaild(const string& usr, const unsigned int& rpn)
{
    for (size_t i = 0; i < usr.length(); i++)
    {
        if (((int)usr[i] - 48) >= rpn && usr[i] != '-' && usr[i] !=
         '.')
        {
            return false;
        }
    }
    return true;
}

void print(char* dpnArr, double decimal, size_t arrLen, const string&
 usr, const unsigned int& dPN)
{
    cout << usr << "的" << dPN << "进制表示为: ";
    if (dpnArr != nullptr)
    {
        if (dPN == 16)
        {
            cout << "0x";
        }
        size_t spcMrk = 0;

        for (size_t i = 0; i < arrLen; i++)
        {
            if (i == 1)
            {
                if (usr[0] == '-')
                {
                    cout << " ";
                    spcMrk = 0;
                }
            }
            if (spcMrk % 4 == 0 && spcMrk != 0 && dpnArr[i] != '.')
            {
                cout << " ";
            }
            if (dpnArr[i - 1] == '.')
            {
                spcMrk = 0;
            }
            cout << dpnArr[i];
            spcMrk++;
```

```
        }
        cout << endl;
    }
    else
    {
        cout << decimal << endl;
    }
    return;
}

int main()
{
    while(true) {
        system("reset");
        cout << "--------------------进制转换器--------------------" <<
         endl;

        cout << "请选择原始进位制：" << endl;
        cout << "（输入0退出程序）" << endl;
        cout << "1. 二进制" << "\t2. 八进制" << "\t3. 十进制" << "\t4.
         十六进制" << endl;
        cout << "请选择- ";
        unsigned int rPN = 0;
        cin >> rPN;
        switch (rPN)
        {
            case 1:
                rPN = 2;
                break;
            case 2:
                rPN = 8;
                break;
            case 3:
                rPN = 10;
                break;
            case 4:
                rPN = 16;
                break;
            case 0:
                cout <<
                 "--------------------程序已结束--------------------"
                 << endl;
                exit(0);
                break;
            default:
                cout << endl;
                cout << "⚠️ 输入有误！" << endl;
                cout << endl;
                if (cin.fail())
                {
```

```
            cin.clear();
            cin.ignore(INT_MAX, '\n');
            cout << "🧭 按下回车键继续...";
            cin.get();
        }
        continue;
}

cout << endl;
cout << "请输入该" << rPN << "进制数（含符号）：" << endl;
cout << "请输入- ";
string usrInput = "";
cin >> usrInput;
if (!isVaild(usrInput, rPN))
{
    cout << endl;
    cout << "⚠️ 输入有误！" << endl;
    cout << endl;
    cin.ignore(INT_MAX, '\n');
    cout << "🧭 按下回车键继续...";
    cin.get();
    continue;
}

cout << endl;
cout << "请选择目标进位制：" << endl
<< "1. 二进制" << "\t2. 八进制" << "\t3. 十进制" << "\t4.
 十六进制" << endl;
cout << "请选择- ";
unsigned int dPN = 0;
cin >> dPN;
switch (dPN)
{
    case 1:
        dPN = 2;
        break;
    case 2:
        dPN = 8;
        break;
    case 3:
        dPN = 10;
        break;
    case 4:
        dPN = 16;
        break;
    default:
        cout << endl;
        cout << "⚠️ 输入有误！" << endl;
        cout << endl;
```

```
            if (cin.fail())
            {
                cin.clear();
                cin.ignore(INT_MAX, '\n');
                cout << "🧭 按下回车键继续...";
                cin.get();
            }
            continue;
        }

        size_t arrLen = 0;
        char* dpnArr = nullptr;
        double decimal = 0;

        if (dPN == 10)
        {
            decimal = DpnToDEC(usrInput, (int)usrInput.length(), rPN);
        }
        else if (rPN == 10 && dPN != 10)
        {
            dpnArr = DecToDPN(usrInput, arrLen, dPN);
        }
        else
        {
            decimal = DpnToDEC(usrInput, (int)usrInput.length(), rPN);
            dpnArr = DecToDPN(to_string(decimal), arrLen, dPN);
        }

        cout << endl;
        print(dpnArr, decimal, arrLen, usrInput, dPN);
        delete [] dpnArr;

        cout << endl;
        cout << "✅ 转换成功，是否继续? " << endl << endl
             << "1- 重新开始" << "\t2- 退出程序" << endl;
        cout << "请选择- ";
        int isExit = 0;
        cin >> isExit;
        if (isExit != 1)
        {
            break;
        }
    }
    cout << "--------------------程序已结束--------------------" <<
     endl;
    return 0;
}
```