

NET2006: Object-Oriented Programming

Input Solutions

Simple Input Function

Ensure that your program continues to operate correctly when a user enters non-numeric characters while inputting integer values.

```
#include <iostream>
using namespace std;

void Input(int& rNumToInput)
{
    cin >> rNumToInput; // reference variable gives direct access

    while ( ! cin.good() ) { // check for bad input
        cin.clear(); // reset status flags in cin object
        cin.ignore(128, '\n'); // empty keyboard buffer of characters
        cout << "Invalid entry.\nPlease enter again.";
        cin >> rNumToInput;
    }
}

void main()
{
    int nAge;
    cout << "What is your age: ";
    Input(nAge); // instead of cin >> nAge;
    cout << "Your age in days is " << nAge * 365;
}
```

Read declaration backwards
"rNumToInput is a reference to an int"

Overloaded again.

Send range using
pass-by-value.

An Even Fancier Input Function

```
#include <iostream>
using namespace std;

void Input(int& rNumToInput, char* pszPrompt, int nLow, int nHigh)
{
    cout << pszPrompt;
    cin >> rNumToInput;

    while ( ! cin.good()
            || rNumToInput < nLow
            || rNumToInput > nHigh ) {
        cin.clear();
        cin.ignore(128, '\n');
        cout << "Invalid entry.\nEnter number between "
              << nLow << " and " << nHigh << endl;
        cout << pszPrompt;
        cin >> rNumToInput;
    }
}

void main()
{
    int nAge;
    Input(nAge, "What is your age: ", 16, 100); // instead of cin >> nAge;
    cout << "Your age in days is " << nAge * 365;
}
```

A Fancier Input Function

```
#include <iostream>
using namespace std;

void Input(int& rNumToInput, char* pszPrompt)
{
    cout << pszPrompt;
    cin >> rNumToInput;

    while ( ! cin.good() ) { // check for bad input
        cin.clear();
        cin.ignore(128, '\n');
        cout << "Invalid entry.\nPlease enter again!";
        cout << pszPrompt;
        cin >> rNumToInput;
    }
}

void main()
{
    int nAge;
    Input(nAge, "What is your age: "); // instead of cin >> nAge;
    cout << "Your age in days is " << nAge * 365;
}
```

Overloaded function. Same function
name; different arguments.

Read declaration backwards
"pszPrompt is a pointer to a char"

Adapting to Other Data Types

```
#include <iostream>
using namespace std;

void Input(double& rdNumToInput, char* pszPrompt)
{
    cout << pszPrompt;
    cin >> rdNumToInput;

    while ( ! cin.good() ) {
        cin.clear();
        cin.ignore(128, '\n');
        cout << "Invalid entry.\nPlease enter again!";
        cout << pszPrompt;
        cin >> rdNumToInput;
    }
}

void main()
{
    double dFahrenheit;
    Input(dFahrenheit, "Enter Current Temperature: ");
    cout << "Celcius: " << (dFahrenheit - 32.0) * 5.0 / 9.0;
}
```

Overloaded again.

Inputting a Three-Part Date

```
#include <iostream>
using namespace std;

void main()
{
    int nYear, nMonth, nDay;
    char cDummy;
    cout << "Enter Date: ";
    cin >> nYear >> cDummy >> nMonth >> cDummy >> nDay;
}
```

Terminates input after finding first non-numeric character.

Clears character from keyboard buffer which caused termination. Retrieved character is thrown away.

Enter Date: 2006-11-17

Inputting Multi-Word String of Characters

```
#include <conio.h> // for getch(), putchar()
#include <ctype.h> // for isprint()
#include <iostream>
using namespace std;

void Input(char* pszInput, int nMaxNumCharacters)
{
    int i = 0;
    char c;
    while ((c = getch()) != '\r') {
        if (c == '\b' && i > 0) { // backspace
            -i; // decrement index into char array, next input overwrites old
            putchar('\b'); // output backspace -- moves cursor backwards
            putchar(' '); // overwrite old screen character to erase visual
            putchar('\b'); // output backspace -- moves cursor backwards
        }
        else if (isprint(c) && i < nMaxNumCharacters) {
            putchar(c); // echo character to screen (not done by getch())
            pszInput[i] = c; // capture keyboard character to char array
            ++i; // next index position -- get ready for next char
        }
    }
    pszInput[i] = '\0'; // place "end-of-string" flag after last input character
}
```

Input keyboard character – assign to

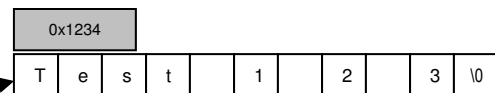
Terminates input on <Enter>

```
void main()
{
    const int INPUT_BUFFER_LENGTH = 10;
    char szInput[INPUT_BUFFER_LENGTH+1]; // create array of char
    cout << "Enter text with embedded blanks: ";
    Input(szInput, INPUT_BUFFER_LENGTH); // send address of array
    cout << endl << szInput;
}
```

Important Note: Array name without square brackets means address of.

Enter text with embedded blanks: Test 1 2 3

Memory Map: main()



Memory Map: Input()

