Валидация на входа

При въвеждане на входа от потребителя често се налага да проверим дали потребителят е спазил нужните ограничения за входа. Тази проверка наричаме валидация на входа. Разгледайте следните примери:

1. Да се състави програма, която въвежда елементите на целочислен масив с големина n, n <=1024, и после ги извежда.
2. Да се състави програма, която въвежда с число месеца на раждане на потребителя и извежда символната му репрезентация.

- **Assert**

assert(<условие>), където:

assert е запазена дума;

<*условие*> е булево условие.

Ако условието е изпълнено, изпълнението на програмата продължава, както обикновено. Ако условието не е изпълнено, изпълнението на програмата се прекратява със съобщение за грешка. Функцията се намира в хедъра cassert (или assert.h), което предполага включването му.

Пример 1:

```
#include <iostream>
#include <cassert>

int main()
{
    int n;
    std::cin >> n;
    assert(n >= 1 && n <= 1024);

    int arr[1024];

    for(int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    for(int i = 0; i < n; i++) {
        std::cout << arr[i];
    }
    return 0;
}

//Input: 5     1 2 3 4 5
//Output: 12345

//Input: 1025
//Output: a.out: main.cpp:16: int main(): Assertion `n >= 1 && n <= 1024' failed.
```

Пример 2:

```
#include <iostream>
#include <cassert>

int main()
{
    int n;
    std::cin >> n;
    assert(n >= 1 && n <= 12);

    switch(n) {
        case 1: std::cout << "January"; break;
        case 2: std::cout << "February"; break;
        case 3: std::cout << "March"; break;
        case 4: std::cout << "April"; break;
        case 5: std::cout << "May"; break;
        case 6: std::cout << "June"; break;
        case 7: std::cout << "July"; break;
        case 8: std::cout << "August"; break;
        case 9: std::cout << "September"; break;
        case 10: std::cout << "October"; break;
        case 11: std::cout << "November"; break;
        case 12: std::cout << "December"; break;
    }
    return 0;
}

//Input: 1
//Output: January

//Input: -1
//Output: a.out: main.cpp:16: int main(): Assertion `n >= 1 && n <= 12' failed.
```

- **Изключения (exceptions)**

```
try {
        if (<условие>) {
                throw <some_exception_message>;
        }
} catch (const char* exception) {
        <логика_която_се_справя_с_изключението>
}
```
, където:

try, *catch* са запазени думи;
<*условие*> е булев израз;
<*some_exception_message*> е съобщение за грешка;
<*логика_която_се_справя_с_изключението*> е оператор или множество от оператори.

Как работи горният фрагмент? В <условие> записваме условието, при което входът е невалиден.

- Ако условието е изпълнено, програмата влиза в тялото на условния оператор и предизвиква грешка чрез ключовата дума throw. Съобщението на тази грешка е съобщението, записано в <some_exception_message>. След това програмата влиза в catch и изпълнява <*логика_която_се_справя_с_изключението*>.
- Ако условието не е изпълнено, т.е. входът е валиден, програмата продължава изпълнението си след края на тялото на if оператора, без въобще да влиза в него.

Изключенията често са предпочитаният начин за валидация на входа. Обикновено когато използваме изключения за валидация, програмата ни придобива следния вид:

```
try {
    // въвеждане на входа

    if (!<условие_за_валидност_на_входа>) {
      throw "Invalid input!"; // или друго съобщение
    }

    <същинската_логика>;
} catch (const char* exception) {
    std::cout << exception << std::endl;
 }
```

Пример 1:

```
#include <iostream>

using namespace std;

int main()
{
    try {
        int n;
        std::cin >> n;

        if (n < 1 || n > 1024) {
            throw "Invalid value for n.";
        }

        int arr[1024];
        for(int i = 0; i < n; i++) {
                std::cin >> arr[i];
        }

        for(int i = 0; i < n; i++) {
                std::cout << arr[i];
        }
    } catch(const char* exception) {
        std::cout << exception << std::endl;
    }

    return 0;
}
//Input: 0
//Output: Invalid value for n.

//Input: 1 1
//Output: 1
```

Пример 2:

```
#include <iostream>

int main()
{
    try {
        int n;
        std::cin >> n;
        if(n < 1 || n > 12) {
            throw "Invalid value for n.";
        }

        switch(n) {
            case 1: std::cout << "January"; break;
            case 2: std::cout << "February"; break;
            case 3: std::cout << "March"; break;
            case 4: std::cout << "April"; break;
            case 5: std::cout << "May"; break;
            case 6: std::cout << "June"; break;
            case 7: std::cout << "July"; break;
            case 8: std::cout << "August"; break;
            case 9: std::cout << "September"; break;
            case 10: std::cout << "October"; break;
            case 11: std::cout << "November"; break;
            case 12: std::cout << "December"; break;
        }
    } catch (const char* exception) {
        std::cout << exception << std::endl;
    }
    return 0;
}

//Input: 12
//Output: December

//Input: 0
//Output: Invalid value for n.
```

- **Въвеждане до първи валиден вход - do...while**

Въвеждаме необходимото за програмата. Ако условието за валиден вход не е изпълнено, позволяваме на потребителя да въвежда отново, докато не въведе валидна стойност.

Пример 1:

```
#include <iostream>

int main()
{
    int n;
    do {
        std::cout << "Enter value for n between 1 and 1024: ";
        std::cin >> n;
    } while (n < 1 || n > 1024);

    int arr[1024];
    for(int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    for(int i = 0; i < n; i++) {
        std::cout << arr[i];
    }
    return 0;
}

//Input: 4 1 2 3 4
//Output: 1234

//Input: 0 10 1 2 3 4 5 6 7 8 9 10
//Output: demands input again first time; second time outputs "12345678910"
```

Пример 2:

```
#include <iostream>

int main()
{
    int n;
    do {
        std::cout << "Enter value for n between 1 and 12: ";
        std::cin >> n;
    } while (n < 1 || n > 12);

    switch(n) {
        case 1: std::cout << "January"; break;
        case 2: std::cout << "February"; break;
        case 3: std::cout << "March"; break;
        case 4: std::cout << "April"; break;
        case 5: std::cout << "May"; break;
        case 6: std::cout << "June"; break;
        case 7: std::cout << "July"; break;
        case 8: std::cout << "August"; break;
        case 9: std::cout << "September"; break;
        case 10: std::cout << "October"; break;
        case 11: std::cout << "November"; break;
        case 12: std::cout << "December"; break;
    }
    return 0;
}

//Input: 12
//Output: December

//Input: 0 10
//Output: demands input again first time; second time outputs October
```

- **Условен оператор if**

Въвеждаме необходимото за програмата. Проверяваме дали условието за валиден вход е изпълнено. Ако е, продължаваме работата си с програмата. Ако не е, изписваме съобщение за грешка и прекратяваме изпълнението на програмата с *exit(1)* или *return <invalid_value>*. Този подход се счита за най-малко предпочитан от гореизброените.

Пример 1:

```
#include <iostream>

int main()
{
    int n;
    std::cin >> n;
    if(n < 1 || n > 1024) {
        std::cout << "Invalid value for n.";
        exit(1);
        //return 1;
    }

    int arr[1024];
    for(int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    for(int i = 0; i < n; i++) {
        std::cout << arr[i];
    }
    return 0;
}

//Input: 0
//Output: Invalid value for n.

//Input: 1 1
//Output: 1
```

Пример 2:

```
#include <iostream>

int main()
{
    int n;
    std::cin >> n;
    if(n < 1 || n > 12) {
        std::cout << "Invalid value for n.";
        exit(1);
        //return 1;
    }

    switch(n) {
        case 1: std::cout << "January"; break;
        case 2: std::cout << "February"; break;
        case 3: std::cout << "March"; break;
        case 4: std::cout << "April"; break;
        case 5: std::cout << "May"; break;
        case 6: std::cout << "June"; break;
        case 7: std::cout << "July"; break;
        case 8: std::cout << "August"; break;
        case 9: std::cout << "September"; break;
        case 10: std::cout << "October"; break;
        case 11: std::cout << "November"; break;
        case 12: std::cout << "December"; break;
    }
    return 0;
}

//Input: 12
//Output: December

//Input: 0
//Output: Invalid value for n.
```