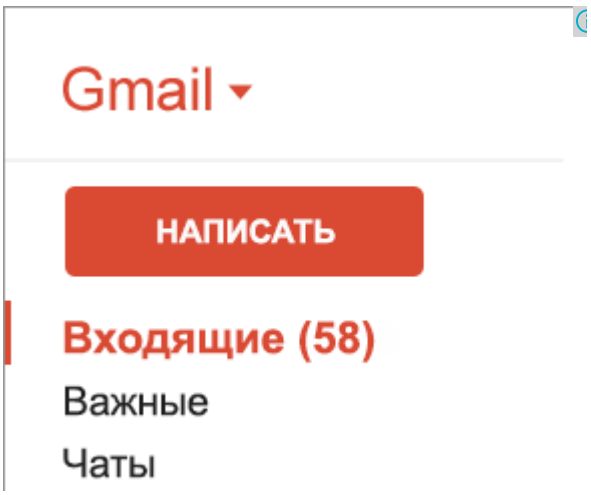
Конструкция switch — case в C++

12-10-21 | root | C/C++ | 10



Сегодня мы научимся пользоваться этой полезной конструкцией языка c++.

Очень часто в процессе написания программы требуется писать длинные if-else конструкции, например, когда мы получаем какой-либо ключ от пользователя; если вы пишете игру, то придется проверять на какую кнопку нажал игрок (вправо, влево, пробел и т.д.).

В этой статье мы узнаем как удобно оформлять подобные конструкции с помощью ***switch case***, а так же узнаем немного о ***enum*** типах, которые хорошо подходят для работы со switch case.

Конструкция switch-case — это удобная замена длинной if-else конструкции, которая сравнивает переменную с несколькими константными значениями, например int или char.

Синтаксис

```
switch ( <переменная> ) {
case значение1:
  Выполнить если <переменная> == значение1
  break;
case значение2:
  Выполнить если <переменная> == значение2
  break;
...
default:
  выполнить, если ни один вариант не подошел
  break;
}
```

Переменная в скобках сравнивается со значениями, описанными после ключевого слова case. После двоеточия находится код, который будет выполнен в случае если переменная оказалась равной текущему значению. break необходим для того, чтобы прервать выполнение switch. Рассмотрим пример, где нет break:

```
int a=1;
switch(a)
{
    case 1:
        a++;
    case 2:
        a++;
    case 3:
        a++;
}
cout<<"a= "<<a;
```

Данная программа выведет a = 4.

Значения для сравнения, описанные после case могут быть только константами, поэтому следующий вариант использования switch-case — неверен:

```
int a = 10;
int b = 10;
int c = 20;

switch ( a ) {
case b:
  // Code
  break;
case c:
  // Code
  break;
default:
  // Code
  break;
}
```

При попытке скомпилировать данную программу, вы получите подобное сообщение:

```
test.cpp:9: error: 'b' cannot appear in a constant-expression
```

Блок default — необязателен, но он полезен для обработки исключительных ситуации.

Следующая программа демонстрирует один из возможных вариантов использования switch-case:

```
#include <iostream>

using namespace std;

void playgame()
{
    cout << "Play game called";
}
void loadgame()
{
    cout << "Load game called";
}
void playmultiplayer()
{
    cout << "Play multiplayer game called";
}

int main()
{
  int input;

```

```
22    cout<<"1. Play game\n";
23    cout<<"2. Load game\n";
24    cout<<"3. Play multiplayer\n";
25    cout<<"4. Exit\n";
26    cout<<"Selection: ";
27    cin>> input;
28    switch ( input ) {
29    case 1:
30      playgame();
31      break;
32    case 2:
33      loadgame();
34      break;
35    case 3:
36      playmultiplayer();
37      break;
38    case 4:
39      cout<<"Thank you for playing!\n";
40      break;
41    default:
42      cout<<"Error, bad input, quitting\n";
43      break;
44    }
45    cin.get();
46  }
```

Эта программа показывает простой способ обработки вводимых пользователем данных.

Минус данной программы в том, что она дает только одну попытку, без права на ошибку 🙂 . Это легко исправить, заключив весь блок switch-case в цикл. Но может возникнуть вопрос: внутри switch-case есть break, не прервет ли он выполнение цикла? Нет, break прервет только выполнение блока switch.

Сравнение switch-case с if-else

Если у вас возникают проблемы с пониманием того, как работает switch-case, посмотрите на следующую if-else конструкцию, она работает точно так же, как и switch

```
1   if ( 1 == input )
2   {
3       playgame();
4   }
5   else if ( 2 == input )
6   {
7       loadgame();
8   }
9   else if ( 3 == input )
10  {
11      playmultiplayer();
12  }
13  else if ( 4 == input )
14  {
15      cout << "Thank you for playing!\n";
16  }
17  else
18  {
19      cout << "Error, bad input, quitting\n";
20  }
```

Если мы можем сделать то же самое с помощью if-else, зачем вообще нужен switch? Главное преимущество этой конструкции в том, что нам понятно, как работает программа: единственная переменная контролирует поведение программы. В случае с if-else, придется внимательно читать каждое условие.

Создаем собственные типы с помощью enumeration

Иногда при написании программ могут понадобится переменные, которые могут принимать только строго определенные значения, которые известны вам заранее. Например, вы можете задать ограниченный набор

цветов, которые пользователь может выбрать. Очень удобно иметь сразу и набор доступных констант и тип переменной, который связан с этим набором. Более того, подобные переменные хорошо подходят для использования в switch-case, так как вы знаете все возможные значения заранее.

Тип **enum** (сокращение от «**enumerated type**«, перечисляемые типы) содержит ***перечисление*** различных значений, например цветов радуги:

```
enum RainbowColor {
    RC_RED, RC_ORANGE, RC_YELLOW, RC_GREEN, RC_BLUE, RC_INDIGO, RC_VIOLET
};
```

Несколько важных моментов:

- Для объявление перечисляемого типа используйте ключевое слово **enum**.
- Имя нового типа вы задаете сами, например RainbowColor.
- Все возможные значения для данного типа перечислены в фигурных скобках.
- После объявления перечисления необходима точка с запятой.

Теперь вы можете объявлять переменные с типом RainbowColor:

```
RainbowColor chosen_color = RC_RED;
```

И, как уже говорилось, эти переменные хорошо подходят для использования в switch:

```
switch (chosen_color)
{
    case RC_RED: /* paint screen red */
    case RC_ORANGE: /* paint screen orange */
    case RC_YELLOW: /* paint screen yellow */
    case RC_GREEN: /* paint screen green */
    case RC_BLUE: /* paint screen blue */
    case RC_INDIGO: /* paint screen indigo */
    case RC_VIOLET: /* paint screen violet */
    default: /* обработка исключений */
}
```

Так как мы используем перечисления, мы можем быть уверенными, что рассмотрели все возможные значения переменной. Значения констант в перечислении — это простой int, по умолчанию каждое следующее значение больше предыдущего на 1. Для первого — 0, для второго — 1 и т.д. В нашем случае: RC_RED = 0 и RC_ORANGE = 1.

Вы также можете задать собственные значения:

```
enum RainbowColor {
    RC_RED = 1, RC_ORANGE = 3, RC_YELLOW = 5, RC_GREEN = 7, RC_BLUE = 9,
    RC_INDIGO = 11, RC_VIOLET = 13
};
```

Преимущество использования перечисляемых типов в том, что вы можете задать имя значениям, которые иначе пришлось бы хард-кодить. Например, если вы пишете игру крестики-нолики, вам нужен способ представления крестиков и ноликов на доске. Это может быть 0 для пустой клетки, 1 для нолика и 2 для крестика. Значит, наверняка придется использовать подобный код:

```
if ( board_position == 1 )
{
    /* do something */
}
```

Данный код очень сложен для понимания и обслуживания, потому что совсем не понятно, что означает цифра 1. Enum типы позволяют избегать таких ситуаций:

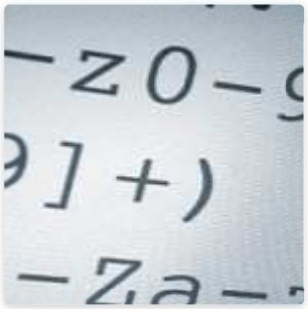
```
enum TicTacToeSquare { TTTS_BLANK, TTTS_O, TTTS_X };

if ( board_position == TTTS_O )
{
    /* some code */
}
```

Теперь все гораздо понятнее 🙂 .

На этом всё! Подписывайтесь и не пропустите новые уроки! 🙂

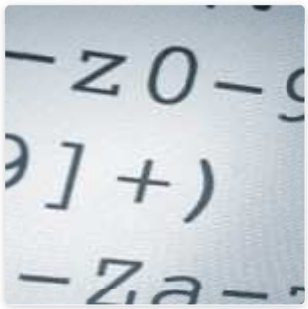
Смотрите также

Структуры в C++



Основы регулярных выражений в Java. Часть 3

Команды для работы по протоколу SSH



Несколько полезных регулярных выражений для веб-разработки.

Функции в C++

Циклы в C++

Перечисляемые типы (enum) в Java

Продвинутое использование cURL в PHP

10 Комментариев

VAN:

10.05.2013 в 8:46 пп

А что, если на вариантах 3 и 5 должно происходить одно и то же?

Не писать же это одно и то же два раза!

Можно ли одним case обслужить несколько вариантов?

Ответить

root:

11.05.2013 в 10:35 дп

Здравствуйте!

Используйте такую конструкцию:

```
switch (i) {
    case 2:
        cout << 1 << endl;
        break;
    case 3:
    case 5:
        cout << "same event\n";
        break;
    case 6:
        cout << 6 << endl;
}
```

Ответить

Игорь:

01.06.2013 в 2:31 пп

Тогда пишешь:

swith (i){

case 3, 5 :

cout << "example";

break;

```
}
```

Ответить

eusi:

08.12.2013 в 10:34 дп

Запятая не работает в этом операторе в Си в среде Dev-C++.

Ответить

Вадим:

28.12.2013 в 7:43 пп

помогите найти ошибку… Все запускаеться, но вложеная програма выполняеться без ввода даных

```
#include «iostream.h»
#include «windows.h»
#include «string.h»
#include «ctype.h»

char* Rus(const char* text);
void main()
{ int men;
cout<<Rus("Програма №1: підрахунок кількості бу
cout<<Rus("Програма №2:Визначення першои букви д
cout<>men;
switch(men)
{case 1:
{
    char stroka[50];
    int b,l,g;
    cout<<Rus("Задайте значення з клавіатури\n"
    cin.getline(stroka,50);
    b=strlen(stroka);
```

```
20      cout<<Rus("Довжина рядка\n");
21      cout<<b<<endl;
22      g=0;
23      for (l=0;l<=b;l++) if(stroka[l]=='a') ++g;
24      cout<<Rus("Кількість букв A\n");
25      cout<<g<<endl;
26      system ("pause");
27  }
28  break;
29
30  case 2:
31      {   char strok[50];
32      int j,x,y;
33      cout<<Rus("Задайте значення з клавіатури\n")
34      cin.getline(strok,50);
35      x=strlen(strok);
36      for (j=0;j<=x;j++) if(strok[j]==' ')break;y=
37      cout<<Rus("Перша буква другого слова\n");
38      cout<< strok[y]<<endl;
39      system ("pause");
40      }   break;
41
42  case 3:
43      {char str[50];
44      int m,i,p;
45      cout<<Rus("Задайте значення з клавіатури\n")
46      cin.getline(str,50);
47      m=strlen(str);
48      cout<<Rus("Довжина рядка\n");
49      cout<<m<<endl;
50      p=0;
51      for (i=0;i<=m;i++) if(str[i]==' ') break; p=
52      cout<<Rus("Довжина слова\n");
53      cout<<p<<endl;
54      system ("pause");
55      }break;
56  default:cout<<Rus(" Невірний формат\n");
57      system ("pause");
58  }
59
60  }
61
62  char bufRus[256];
63  char* Rus(const char* text)
64  {
65      CharToOem(text,bufRus);
66      return bufRus;
67  }
```

Ответить

vasso Picasso:

04.01.2014 в 12:57 дп

Здорова.
«coutmen;» измени на «cin>>men», ты пишешь вместо чтения.

И далее возможны проблемы с использованием «cin.getline(stroka,50);»,
может понадобиться «cin.ignore();» перед ним. не уверен …

Лучше делай cin.getline(stroka, sizeof(stroka));

Ответить

ErrorER:

30.03.2014 в 10:46 дп

а строки можно после case ставить?

Ответить

root:

30.03.2014 в 3:46 пп

Здравствуйте!

К сожалению, возможности использовать оператор switch-case со строковыми константами нет.

Ответить

Shiny:

19.08.2014 в 11:06 дп

Большое спасибо! Все вроде бы понятно. Хм интересно почему нельзя поставить переменные после case

Ответить

Рената:

12.04.2015 в 12:50 пп

Помогите, пожалуйста, что не так? Нужно с помощью swithc-case позволить делать выбор между умножением и вычитанием:

```
1  #include "stdafx.h"
2  #include
3  #include
4  #include
5  #include
6
7  #define d1 10
8  #define d2 100
9  #define N 10
10
11 int main()
12 {
13     setlocale(LC_ALL, "Russian");
14     int numb1, numb2, correctAnswer, answer, c
15     char reset;
16     printf("Умножение или вычитание? y/n \n");
17         fflush(stdin);
18         scanf("%c", &amp;reset);
```

```
19
20          switch(reset)
21          {
22          case 'y':
23              {
24
25
26
27
28      char reset = 1;
29      while(reset)
30      {
31          numb1 = 0;
32          numb2 = 0;
33          correctAnswer = 0;
34          answer = 0;
35          counter = 0;
36
37          for(int i = 0; i &lt; N; ++i)
38          {
39              srand(time(NULL));
40              numb1 = d1 + rand() % d2;
41              numb2 = d1 + rand() % d2;
42              correctAnswer = numb1 - numb2;
43
44              printf(&quot;%d - %d = &quot;, numb
45              scanf(&quot;%d&quot;, &amp;answer);
46              if(answer == correctAnswer)
47              {
48                  printf(&quot;Correct!\n&quot;);
49                  ++counter;
50              }
51  else
52              {
53                  printf(&quot;Incorrect! &quot;)
54                  printf(&quot;%d - %d = %d\n&quo
55              }
56          }
57
58          printf(&quot;Correct answers qty: %d\n&
59          printf(&quot;Grade: &quot;);
60          break;
61      }
62          case &#039;n&#039;:
63              {
64
65      {numb1 = 0;
66          numb2 = 0;
67          correctAnswer = 1;
68          answer = 1;
69          counter = 1;
70
71          for(int i = 0; i &lt; N; ++i)
72          {
73              srand(time(NULL));
74              numb1 = d1 + rand() % d2;
75              numb2 = d1 + rand() % d2;
76              correctAnswer = numb1 * numb2;
```

```
77
78              printf(&amp;quot;%d * %d = &amp;quot;, numb
79              scanf(&amp;quot;%d&amp;quot;, &amp;amp;answer);
80              if(answer == correctAnswer)
81              {
82                  printf(&amp;quot;Correct!\n&amp;quot;);
83                  ++counter;
84              }
85  else
86              {
87                  printf(&amp;quot;Incorrect! &amp;quot;)
88                  printf(&amp;quot;%d - %d = %d\n&amp;quo
89              }
90          }
91
92          printf(&amp;quot;Correct answers qty: %d\n&
93          printf(&amp;quot;Grade: &amp;quot;);
94          break;
95      }
96
97          {
98          case 10:
99              {
100                 printf(&amp;quot;A\n&amp;quot;);
101                 break;
102             }
103         case 9:
104         case 8:
105             {
106                 printf(&amp;quot;B\n&amp;quot;);
107                 break;
108             }
109         case 7:
110         case 6:
111             {
112                 printf(&amp;quot;C\n&amp;quot;);
113                 break;
114             }
115         default:
116             printf(&amp;quot;D\n&amp;quot;);
117         }
118
119 ex0:
120 printf(&amp;quot;Would you like to repeat? y/n \n&amp
121         fflush(stdin);
122         scanf(&amp;quot;%c&amp;quot;, &amp;amp;reset);
123
124         switch(reset)
125         {
126         case &amp;#039;y&amp;#039;:
127             {
128                 break;
129             }
130         case &amp;#039;n&amp;#039;:
131             {
132                 return 0;
133                 break;
134             }
```

```
135             default:
136                 goto ex0;
137             }
138         }
139
140         return 0;
141     }
```

Ответить

Добавить комментарий

Ваш e-mail не будет опубликован. Обязательные поля помечены *

Имя *

E-mail *

один + 1 =

Комментарий

Отправить комментарий