ХИМИКОТЕХНОЛОГИЧЕН И МЕТАЛУРГИЧЕН УНИВЕРСИТЕТ – СОФИЯ

ИНФОРМАТИКА

част първа

Основи на програмирането на C++

лектор: гл. ас. д-р Стефан М. Панов

Катедра “Информатика”

Лекция 5

Условни инструкции if и switch в C++

Съставна инструкция, наричана още блок с код, представлява група инструкции затворени в къдрави (фигурни, големи) скоби.

Нормално една C++ програма изпълнява една инструкция, след което преминава към следващата инструкция и т.н. **Понякога е необходимо последователното изпълнение на инструкциите да бъде нарушено** и една група инструкции да бъде изпълнена при едни условия, а друга група инструкции при други условия. Или пък **група инструкции да бъде изпълнена многократно** преди по-нататъшно изпълнение на програмата. За тая цел езикът **C++ предоставя две категории инструкции**: **условни инструкции** (инструкции за избор) и **инструкции за цикъл**.

Инструкция if

Инструкцията if позволява да направим избор между два изпълними клона на програмата.

Синтаксисът ѝ е следният (на български ако ... иначе):

```
if (управляващ-израз)
{
    if-последователност от инструкции
}
else
{
    else-последователност от инструкции
}
```

! Else частта не е задължителна

Ако управляващият израз при изчисление дава стойност ИСТИНА (true), ще се изпълни if-последователността от инструкции; в противен случай се изпълнява else-последователността от инструкции (ако такава съществува).

Никога не се изпълняват **и двете** последователности. **Управляващият израз** може да е от произволен тип, действителен за C++-изрази. Главното е резултатът от неговото изчисление да може да се тълкува като стойност ИСТИНА или ЛЪЖА (**true** или **false**).

Да разгледаме пример за използването на if-инструкцията в програма, представляваща опростена версия на играта “Отгатни магическото число”. Програмата генерира случайно цяло число и ви предлага да го отгатнете. Ако успеете, се извежда съобщение ** Правилно **. В тая програма е показана една библиотечна **функция rand(),** която **връща случайно избрано цяло положително число**. За използването на тази функция е необходимо да се включи в програмата заглавния файл **<cstdlib>** (или **stdlib.h)**.

```
// Progama "Otgatni magicheskoto chilso ".
#include <iostream> // nashata 9-a programa
#include <cstdlib>
using namespace std;
int main ()
{
      int magic; // magicheskoto chilso
      int guess; // variant na potrbitelia
      magic = rand(); // Poluchavame magicheskoto chilso.
      cout << "Vavedete vashia variant na magicheskoto chislo: ";
      cin >> guess;
      if (guess == magic) { cout << "\n** Pravilno **" ; }
      return 0;
}
```

Проверката в управляващия израз става става с оператора за отношение “==”.

Когато **if-последователността** или **else-последователността** (понякога се наричат **„тялото** на if или else“) се състои от единична инструкция къдравите скоби могат да се пропуснат, т.е. редът:

```
if (guess == magic) { cout << "\n** Pravilno **" ; }
```

може да се запише и така:

```
if (guess == magic) cout << "\n** Pravilno **" ;
```

! Препоръчително е винаги да използвате къдрави скоби, даже когато тялото на if или else се състои от единична инструкция.

Нека усъвършенстваме нашата програма и в нейната нова версия да включим **else** част за извеждане на съобщение, че предположението на потребителя се е оказало погрешно. Добавяме в програмата следния код:

```
else { cout << "\nSyjaliavame, no sgreshihte."; }
```

! Въпрос на лично предпочитание е дали отварящата кудрава скоба след if или else ще се напише на същия ред или на нов. Т.е. кое да предпочетем:

if (управляващ-израз)
{
или

if (управляващ-израз) {

Важно за управляващия израз (условието) на if

Типът на управл. израз не е задължително да се ограничава до оператори за отношения и логически оператори или операнди от типа bool. Главното е резултатът от изчислението на условието да може да се тълкува (интерпретира) като стойност ИСТИНА или ЛОЖА.

Да си припомним, че 0 автоматично се преобразува във **false**, а **всички** ненулеви стойности— в **true**. Това означава, че произволен израз, който дава резултат нулева или ненулева стойност, може да се ползва за управление на if-инструкция. Например, следващата програма прочита две цели числа от клавиатурата и извежда на екрана частното от делението на първото на второто. За да не допуснем деление на нула е добавена if-инструкция.

```
#include <iostream> // Delim pyrwoto chislo na vtoroto.
using namespace std; // nashata 10-a programa
int main() {
    int a, b;
    cout << "Vyvedete  dve celi chisla: ";
    cin >> a >> b;
    if (b) { cout << (float)a/b << "\n"; }
    else {
```

```
        cout<< "Na nula ne triabva da se deli!\n";
    }
    return 0;
}
```

Обърнете внимание, че стойността на променливата **b** (делителя) се сравнява с нула чрез инструкцията **if (b)**, а не с инструкция **if (b ! = 0)**. Последната също е вярна, но не е необходима и не е така ефективна.

Аналогично за казаното по-рано за cout:

В общия случай една инструкция за въвеждане може да съдържа произволно количество оператори за въвеждане „>>” като след всеки следва променливата, на която искаме да зададем стойност.

В тоя случай за разделител между променливите при може да се ползва интервал, хоризонтална табулация или „Enter” за нов ред.

Вложени if-инструкции

Вложени if-инструкции се образуват в случай, когато вътре в if или else блока на една if инструкция се използва друга if-инструкция. Вложените if-инструкции са много използвани в програмирането.

Главното тука е да се помни, че: else-частта винаги се отнася към най-близката if-инструкция, намираща се вътре в същия програмен блок, но още несвързана с никоя друга else- инструкция. Пример:

```
if(i) {
      if(j) {statement1;}
      if(k){ statement2;} // Тая if-инструкция е
      else {statement3;} // свързана с тая else-инструкция.
}
else{ statement;} // Тая else-инструкция е свързана с if(i).
```

Както се твърди в коментарите, последната else-инструкция не е свързана с инструкцията if (j), **защото те не се намират в един блок** (независимо от факта, че if (j)-инструкцията е най-близката, която си няма “else-другарче”). Вътрешната else-инструкция е свързана с инструкцията if (к), защото тя е най-близката и се намира вътре в същия блок.

!! Препоръчва се инструкциите от if или else блока да започват по-вдясно от ключовата дума if или else. Това силно улеснява читаемостта на програмата. Същото правило важи и при другите инструкции, където се използват блокове, например инструкциите за цикъл.

Ще демонстрираме употребата на вложени инструкции посредством поредното усъвършенстване на програмата “Отгатни магическото число” (тук играчът получава реакция от програмата при неправилен отговор).
Ще покажем само новият вид на if инструкцията, останалата част от програмата е без промени в сравнения с предишния вариант.

```
if (guess == magic) { // chast ot nashata 11-a programa
      cout << "** Pravilno **\n";
      cout << "Imenno " <<magic << " e magicheskoto chislo.\n";
}
else {
      cout << "Syjaliavame, no sgreshihte.\n ";
      if(guess > magic) {
                  cout << "Vashiat izbor previshava magicheskoto chislo.\n";
      }
      else {
            cout << "Magicheskoto chislo previshava vashiat izbor.\n";
      }
}
```

if-else-if стълбица

Много разпространена конструкция в програмирането, в основата на която се намира вложена if-инструкция, се явява “стълбицата” if-else-if. Тя може да се представи в следния вид:

```
if(условие1)
    {if-блок1}
else
    if(условие2)
        {if- блок 2}
    else
        if(условие3)
            {if- блок 3}
        else
            ...
                else
                    {else- блок }
```

Тая конструкция често се записва по-сбито по следния начин:

```
if(условие1)
      {if-блок1}
else  if(условие2)
      {if-блок 2}
else if(условие3)
      {if-блок 3}
...
else
      {else-блок }
```

При if-else-if стълбицата се изпълнява само if-блока на първото условие, което е ИСТИНА. Ако нито едно от условията не е ИСТИНА, се изпълнява else-блока на завършващата else-част. Ако липсва завършваща else-част, не се изпълнява нито един от блоковете на if-else-if стълбицата.

Работата на if-else-if-стълбицата е показана в следващата програма.

```
#include <iostream> // nashata 12-a programa
using namespace std; // demonstracia na izpolzvaneto na stylbica if-else-if
int main() {
	int x;
	cout << "Vavedete malko cialo polojitelno chislo. : ";
	cin >> x;
	if(x==1) {cout << "x e ravno na edno.\n";}
	else if (x==2) {cout << "x e ravno dve.\n ";}
	else if (x==3) {cout << "x e ravno na tri.\n ";}
	else if (x==4) {cout << "x e ravno na chetiri.\n";}
	else if (x==5) {cout << "x e ravno na pet.\n";}
	else {cout << "x ne popada v diapazona ot 1 do 5.\n ";}
	return 0;
}
```

Условна операция

Това е единствената тринарна (с три операнда) инструкция в C++. Може да се разглежда като разновидност на **if-else**. Тя има следния синтаксис:

операнд1 ? операнд2 : операнд3

Ако стойността на операнд1 е различна от нула се изчислява операнд2 и неговата стойност става резултат от цялата операция, в противен случай се изчислява операнд3 и неговата стойност става резултат от цялата операция.

```
#include <iostream> // primer za
using namespace std; // uslovna operacia
int main() { // nashata 12b programa
    int x=10, y=20, z;
    z = (x > y) ? x : y;
    cout << "z = " << z << "\n";
    return 0; }
```

Инструкция switch

Инструкцията switch служи за отклонение на програмата по едно от няколко възможни разклонения в зависимост от стойността на целочислен управляващ израз. Той има следния синтаксис:

```
switch (управляващ-израз) {
case константа1:  инструкции1
      break;
case константа2: инструкции2
      break;
.....
case константаN: инструкцииN
      break;
default:default-инструкции
      break;
}
```

където **управляващ-израз** **може да бъде само израз, който връща цяло число (или символна стойност) като резултат**.

константа1,...константаN

може да бъде само **целочислен константен израз**.

инструкции 1 / инструкции 2..../ инструкции N

може да бъде всяка C++ инструкция, включително и друга switch инструкция.

Ако стойността на целочисления управляващ-израз съвпада със стойността на някоя от целочислените константи константа1,...константаN, програмата преминава към изпълнение на инструкциите, които се намират след нея. При достигане на инструкция **break** програмата излиза от тялото на инстр. **switch** и преминава към изпълнение на първата инструкции след нея.

Операторът default не е задължителен. Ако стойността на **целочисления** управляващ-израз не съвпада със стойността на някоя от целочислените

константи константа1,...константаN и секцията default липсва, инструкцията switch се прескача и програмата преминава към изпълнение на първата инструкция след switch.

При употреба на switch-инструкция е необходимо да се знае следното:

- Инструкцията switch се различава от инструкция if по това, че switch-изразът се проверява само с използване на условието за равенство (т.е. на съвпадение на switch-израза със зададените case-константи), докато условният if израз (управляващият израз) може да бъде от произволен тип.
- Никакви две case-константи в една switch-инструкция не могат да имат еднакви стойности.
- switch обикновено е по-ефективна от вложени if-инструкции.
- Последователността от инструкции, свързана с всеки case-клон, не е блок. Обаче пълната switch-инструкция определя блок.

- Ако в някоя от case-константите е пропуснато да се добави инструкция break след поредицата инструкции, принадлежащи на същата константа, програмата продължава с изпълнението на инструкциите на следващата case-константа. Това продължава, докато се срещне break или се стигне до затварящата къдрава скоба на инструкция switch. Да илюстрираме казаното с **пример на следващата страница**. Ако нашият избор е 2 на екрана ще видим:

Po-malko ot 3

Po-malko ot 4

Po-malko ot 5

Както е видно по резултатите, ако инструкция break отсъства в един case-клон се изпълняват и инструкциите, отнасящи се до следващия case-клон.

- **Когато case-константи се добавят една след друга един и същ код ще се изпълни при различни стойности на управляващия израз**. Пример на стр. 23

```
#include <iostream> // demonstracia na poredica ot case-ove bez break
using namespace std; // nashata 13-a programa
int main(){
	int choice;
	cout << "Vavedete cialo v intervala  0-4: "; cin >> choice;
	switch(choice) {
	case 0: cout << "Po-malko ot 1\n";
	case 1: cout << "Po-malko ot 2\n";
	case 2: cout << "Po-malko ot 3\n";
	case 3: cout << "Po-malko ot 4\n";
	case 4: cout << "Po-malko ot 5\n";
		break;
	default:
		cout << "Vashiat izbor e izvyn intervala 0-4\n";
		break;
	}
	return 0; }
```

```
#include <iostream> //  nashata 14-a programa
using namespace std; // demonstracia na poredica ot case-ove izpolzvashti
int main(){                  // edna i syshta posledovatelnost ot instrukcii.
      int choice;
      cout << "Vavedete cialo v intervala  1-3: ";
      cin >> choice;
      switch(choice) { // pri vavejdane na 1, 2, ili 3 se pravi edno i syshto
      case 1:
      case 2:
      case 3:
            cout << "Vashiat izbor e mejdu 1 i 3.\n";
            break;
      default:
            cout << "Vashiat izbor e izvyn posochenia interval.\n";
            break;
      }
      return 0;}
```

Ще завършим лекцията с по-сложен пример за използване на switch и if.

```
#include <iostream> // Primer za programa s izpolzvane na switch i if instrukcii
#include <cmath> // toia fail e neobhodim za funkciite sqrt() I fabs()
using namespace std;  // nashata 15-a programa
const double PI = 3.14159; // obiaviavane na konstantata pi
int main(){
    int choice; // nashiat izbor
    float a;  // vhodna promenliva - radius ili strana na geometrichna figura
    float h; // visochina na ravnostrannia triygylnik
    float r; // sydyrja rezultata ot izchisleniata
    cout << "Vavedete cialo v intervala  1-4 za :\n\n";
    cout << "1. Izchisliavane na lice na okryjnost s raduis a:\n";
    cout << "2. Izchisliavane na lice na kvadrat sys strana a:\n";
    cout << "3. Izchisliavane na lice na ravnostranen triygylnik sys strana a:\n";
    cout << "4. Izchisliavane na obem na sfera s radius a:\n\n";
    cout << "Vashiat izbor :"; cin >> choice;
```

```
if (choice >=1 && choice <=4) { // ako choice e izvyn obhvata
      cout << "a=";                    // niama smisyl da vavejdame a
      cin >> a;   // ako a e otricatelno podsiguriavame da stane + chislo
      a = fabs(a); // fabs() vryshta absoliutnata stoinost na realno chislo
}
switch(choice) {
case 1: r = PI*a*a;
    cout <<"Liceto na okryjnostta s raduis " << a << " e ravno na "<< r <<"\n";
    break;
case 2: r = a*a;
    cout <<"Liceto na kvadrat sys strana " << a << " e ravno na "<< r <<"\n";
    break;
case 3: h = sqrt(a*a - (a/2)*(a/2));
          r = a*h/2;
         cout << "Liceto na ravnostranen triygylnik sys strana " << a
           << " e ravno na "<< r << "\n";
     break;
```

```
        case 4: r = 4*PI*a*a*a/3;
                cout <<"Obemyt na sfera s radius " << a << " e raven na "<< r <<"\n";
                 break;
        default:
              cout << "Vashiat izbor e izvyn posochenia interval 1-4.\n";
              break;
        }
        return 0;
}
```

За по-лесно разучаване на кода коментарите и стойностите на символните низове са дадени в отделни цветове, а всички ключови думи – с удебелен шрифт. Програмата предлага изчисляване на лице или обем на геометрична фигура, като изборът се оставя на потребителя. Самата реализация на избора става с инструкция **switch**. Разбира се, очаква се читателят да е

запознат с формулите за лице на окръжност, квадрат и триъгълник, както и обем на сфера.

Във всяко от изчисленията има и един входен параметър, който се съхранява в променливата **a**. Самото въвеждане на a има смисъл само когато изборът на потребителя е една от предложените формули, което е подсигурено с **if** инструкция. Понеже a може да бъде произволно реално число за нея е избран типът **float**. Тъй като от гледна точка на изискванията на геометрията **a** трябва да бъде положително число, това е подсигурено с помощта на готовата функция **fabs()**, която връща абсолютната стойност на реално число. Вместо извикването на **fabs()**, т.е. a = fabs(a);

същото може да се постигне със следния **if**:

```
if (a<0) {
   a = -a;
}
```

В случай, че изборът е да се изчисли лице на равностранен триъгълник е необходимо първо да се намери височината на триъгълника. Това е направено чрез използването на питагоровата теорема, като формулата може да се опрости (опитайте сами), което пък би направило програмата по-ефективна .

В програмата е използвана още една стандартна функция от C++: **sqrt()**. Тя връща стойността на квадратен корен на своя аргумент. Аргументът ѝ трябва да има тип **double**. Именно поради това при извикване на **sqrt()** изчислената стойност на израза a*a - (a/2)*(a/2), явяващ се аргумент (входен параметър) на функцията, неявно се преобразува към типа **double**. Самата функция също връща стойност от типа **double**. Обърнете внимание на това, че в програмата е включен заглавният файл <cmath>, защото той осигурява поддръжката на функциите **sqrt()** и **fabs()**.

Но откъде знаем, че аргументът и връщаната стойност на **fabs()** са от тип **double**? Да си припомним от лекция 3 що е прототип на функция, за **sqrt()** имаме:

double sqrt(double x);

Именно тоя прототип се съхранява в <cmath>.

Изобщо, в един прототип е дадено всичко необходимо, за да знаем как да извикаме съответната функция!

Важно! C++ поддържа широк набор от математически функции, ето някои от тях: **sin(), cos(), tan(), log(), ceil() и floor()**.

Запомнете, че всички математически функции изискват включването в програмата на заглавния файл <cmath> или <math.h>, като вторият задължително се ползва при езика C.