Ako sa v C++ programuje:

napíšete zdrojový program – je to obyčajný textový súbor (alebo viacero textových súborov),
takýto program preložíte do strojového kódu – kompiluje pomocou kompilátora,
ak počas kompilácie nastala chyba, musíte program opraviť a pokúsiť sa znovu kompilovať,
inak vznikne súbor napríklad *.exe, ktorý môžete spustiť,

Príklad: Program, ktorý zatiaľ nič rozumné nerobí:

```
#include <conio.h> // pouzijeme kniznicu
void main () { // hlavna funkcia
  getch ();
}
```

Napíšte tento program v prostredí BC, uložte si ho na disk pod menom prvý.cpp skompilujte a spustite (stlačte F9).

1. kompilátor prečíta zdrojový program tak, ako človek (zľava doprava a zhora nahol),
2. ak kompilácia dopadla úspešne, na disku vzniknú súbory s názvami prvy.obj a prvý.exe:
3. prvý.obj – dočasný súbor,
4. prvý.exe – sa dá spústiť.

Vysvetlenie programu:

```
#include <conio.h>          // pouzijeme kniznicu
```

kompilátoru povieme, že použijeme príkazy z knižnice conio,
knižnica = zbierka funkcií, premenných, objektov ...
slovo #include môžeme použiť viackrát, ak chceme využívať príkazy z viacerých knižníc.
komentár = text, ktorý kompilátor ignoruje, ale človeku pomáha pochopiť program,
komentár začína značkou // a text za touto značkou sa až do konca riadku ignoruje.

```
void main () {
}
```

definovali sme funkciu s názvom main,
funkcia v C++ neznamená vždy to isté ako v matematike. V C++ ... postupnosť príkazov,

```
void main () = hlavička funkcie,
```

zátvorky { } označujú telo funkcie = príkazy, ktoré sa postupne vykonajú,
špeciálne, funkcia s názvom main = hlavná funkcia, po spustení programu sa začnú vykonávať príkazy tejto funkcie:

```
getch ();
```

príkaz, ktorý čaká na stlačenie klávesu na klávesnici (nefunguje pre klávesy Shift, Ctrl a ešte niektoré

```
)
```

Zhrnutie

Vlastné funkcie definujeme podľa takéhoto vzoru:
typ meno(parametre)

```
{
  telo
}
```

Pričom:
typ ... je typ výsledku: int, float, ... alebo void, v prípade funkcie bez návratovej hodnoty
meno ... je meno funkcie
parametre ... je zoznam parametrov, parametre oddeľujeme čiarkami,
telo ... sú príkazy, ktoré tvoria telo funkcie.
Ak funkcia vracia hodnotu, musíme vo funkcii vždy vykonať príkaz:

```
return výraz;
```

Príkaz return vyhodnotí výraz a jeho hodnota sa stane návratovou hodnotou (výsledkom) funkcie. Príkaz return zá
Pri volaní uvádzame v zátvorkách parametre funkcie:

```
meno(hodnota1, hodnota2, ...);
```

Zásady písania programu:

- bez diakritiky
- malé písmená

-	bez medzier

Niektoré znaky napíšeme držaním pravého Alt a klávesou:

<	*,*
>	*:*
{	*B*
}	*N*
\	*Q*
@	*V*
#	*X*
[	*F*
]	*G*

Komentáre programu_

Int a=5; //jednořádkový komentář

/ a vše, co je za nimi, překladač ignoruje, dokud nenarazí na znaky */*

Hlavičkové súbory

Jsou to soubory, obvykle s příponou .h, které obsahují předdefinované funkce, objekty atp…

Súbor je textový, a obsahuje rôzne funkcie. Ušetrí sa miesto v našom vlastnom programe a môžeme ich používať a volať z iných programov.

Do programu sa vkladajú v hlavicke

#include <soubor.h>.

Obsah hlavičkového souboru head.h:

```
const float PI 3.14159;          /konštanta PI na 6 des.miest
int  objemKoule(int r)           /celociselna funkcia na výpočet objemu s celočíselnou premennou r
{
 return PI*r*r*r;                /výpočet objemu
}
```

Obsah souboru objKoule.cpp

```
#include <head.h>
#include <iostream.h>
cout << objemKoule(10);
```

V souboru objemKoule.cpp už nemusíme deklarovat ani definovat funkci objemKoule a můžeme ji rovnou používat, protože byla definována v hlavičkovém souboru head.h

Premenné

Premenná je ako krabička, ktorá si počas programu pamätá nejakú hodnotu, a môže sa počas programu meniť.
Premenná je pomenované miesto v pamäti, na ktoré si môžeme uložiť nejakú hodnotu – napríklad vstup od užívateľa, príp
pomocnej premennej môžeme počítať, koľkokrát sa vykonala nejaká časť programu.
premenná má v C++ tiež vopred určený typ (celé číslo, znak, …), nemôžeme do premennej uložiť hodnotu iného typu, ako
deklarácii. Premennú deklarujeme príkazom:

typ názov;

Tento príkaz však len vyhradí potrebné miesto v pamäti, nevloží doň žiadnu hodnotu. Preto pred použitím premennej jej n
priradiť (prípadne načítať) hodnotu:

int cislo;
cislo = 42;

char znak;
cin >> znak;

Základné typy premenných:

int – celé číslo
char – znak
string – reťazec (postupnosť znakov)
bool – pravdivostná hodnota (true (1) alebo false (0))
double – desatinné číslo

Typy premenných

typ	**čo do nej môžeme dať**
string	text
1)celočíselné:	
short int	číslo v rozmezí –32768 až 32767
unsigned short int	číslo v rozmezí 0 až 65535
int	číslo v rozmezí –2147483648 až 2147483647
long int	číslo v rozmezí –2147483648 až 2147483647
unsigned long int	číslo v rozmezí 0 až 4294967295
2) reálne (s desatinnou čiarkou):	
float - reálne číslo (má v	proměnné schopné uchovávat velká čísla
double - reálne číslo s d	s plovoucí desetinnou čárkou
long double - reálne číslo najväčšie (má veľkosť 80 bite)	
char	256 celočíselných hodnot, reprezentujících znaky v tabulce ASCII

char znak=65;
cout << znak;

program vypíše znak A (ten je v ASCII 65)
Vytvorenie premenných:

int objem=50,plocha,vyska=7;

čiže je možné vytvoriť aj viac naraz, ale v 1 riadku len jeden typ!
pozor na malé a veľké písmená.

Polia premenných

int zamestnanci[5];

vytvorí pole zamestnanci, ktoré má 5 hodnôt, zamestnanci[0] [1] [2] [3] [4] čiže spolu 5.

int zamestnanci[5]={100,102,200};

Kde přiřadíme proměnným zamestnanci[0] hodnotu 100, zamestnanci[1] hodnotu 102 , zamestnanci[2] hodnotu 200 a ostatním zaměstnancům hodnotu 0. Pamatujte si tedy, že pokud chcete vytvořit pole o 5 proměnných napíšete int zamestnanec[5], ale nejvyšší možný zaměstnanec je zamestnanec[4]

Príklady dobrých deklarácií:

int CeleCislo;

```
int a,b,c,
   aO, a1, a2,
   _1234;
int a,A; // v C a nie je to isté ako A!
Kor korytnacka; // Kor je typ, ktorý vytvorí korytnačku.
```

Príklady zlých deklarácií:

```
int a,b,a; // opakujú sa mená a,
int 1234;  // meno je číslo,
int Suma!; // výkričník,
int void;  // void je vyhradené slovo,
```

Zadávanie údajov do programu v C++e

Robí sa príkazom

cin >> cena; program čaká, kým človek nezadá hodnotu, a tú potom priradí premennej cena.

= !Program to urobí a pokračuje až po klávese Enter!

! Premenná cena musí byť predtým uvedená aj s jej typom v hlavičke programu v časti var!

Ak chceme zadať viac premenných:

cin >> vek >> vyska >> vaha;

Písanie výsledkov a textov

Sa robí príkazom:

cout << "Je ti " << vek << endl;

= program napíše na obrazovku text v úvodzovkách aj s hodnotou premennej vek.

\n nová řádka
\r *návrat na začátek řádky*
\f nová stránka
\t *tabulátor*
\b posun doleva (backspace)
\a krátký zvukový signál
\\ *1 zpětné lomítko*
\´ apostrof
\0 nulový znak
\" uvozovky

cout << "Tohle bude na 1. radku.\n" << "Tohle na druhem" << endl << "A tohle na tretim";

Príklad 0

Výpočty

a=b+c;

Používame znamienka +, -, *, / na numerickej klávesnici, číslice a zátvorky ().

% je modulus, čo je zvyšok po delení.

Znamienka sú:

==	Rovná se	=	Přiřazení hodnoty
!=	Nerovná se	+=	Přiřazení součtu uložené a nové hodnoty
<	Je menší než	-=	Přiřazení rozdílu uložené a nové hodnoty
>	Je větší než		*Pocet=pocet+3;*
<=	Je menší nebo rovná se		je to iste ako
>=	Je větší nebo rovná se		*Počet+=3;*

Príklad 1

Funkcia na zistenie sklačenia klávesy

keypressed;

dáva logickú FALSE dovtedy, kým nestlačíme klávesu (jedno akú). Po stlačení je TRUE.

Musíme v hlavičke zadať knižnicu crt.tpu príkazom uses crt

klavesa:=readkey;

= program čaká, kým nestlačíme klávesu a do premennej klávesa priradí jej kód, premenná musí byť char

!!!tieto funkcie nereagujú na Ctrl, Alt atd.

Náhodný výber

Sa robí príkazom

x = rand()%81;

= za x sa dosadí náhodná hodnota od 0 po 80. !!!

!!!pred prvým príkazom random MUSÍME dať príkaz

srand90;

ktorý zabezpečí náhodnosť pri každom spustení programu!!!

Príklad 2

Myš

Ak chceme v C++e pracovať s myšou, musíme ju spustiť príkazom

mysinic;

Pozíciu myši na ploche zistia

mysx;

mysy;

zistí pozíciu x-ovej a y-ovej súradnice polohy myši

{

telo

}

Vetvenie programu

spôsobí, že program môže ísť rôznymi cestami, podľa nami zadaných podmienok. Dá sa to urobiť:

1. podmienkou if

```
If (c==4)
{
...
}
```

2. podmienkou if ..... Else

```
If (c==4)
{
...
}
else
{
...
}
```

3. Vyber z viac hodnot:

```
switch(cislo)
{
 case 4: cout << "Hezké číslo"; break;
 case 3: cout << "Velmi pěkné!"; break;
 case 2: cout << "Skvělé!";break;
 case 1: cout << "Krásné!";break;
 default: cout << "Číslo mimo interval!";
}
```

A program vždy pokračuje ďalším riadkom ďalej.

Spájanie podmienok

Niekedy potrebujeme použiť viac podmienok naraz, alebo opak podmienky:

!(a<=3)	teda a<3
a>=0 && a<=3	a je od 0 po 3
a>3 || a=3	a je menšie ALEBO rovné 3

And &&
Or ||
Not !

Príklad 3
Príklad 4
Príklad 5

Opakovanie v C++e

Sa robí niektorým z týchto spôsobov:

1. Príkaz *while (x<3)*
   { ak nie je podmienka splnená už na začiatku, príkaz sa nevykoná ani raz
   příkaz;
   }

= kým je podmienka pravdivá (x<3), opakuje sa príkaz A. Ak prestane platiť, program pokračuje príkazom B a ďale

2. Príkaz *do*
   {
   příkazyA;
   B;
   } while (x<3)

= je to isté, len podmienka sa zisťuje na konci vetvenia. Príkazy A a B sa opakujú,
kým platí podmienka x<3. Ak prestane platiť, ide príkaz C.

3. Príkaz f *for (vyraz1;vyraz2;vyraz3)*
   {
   ... *writeln(i);*
   }

Inkrement ++ zvětší hodnotu proměnné o 1
Dekrement -- ju zmensi
Môžeme ich použiť pred premennou aj za premennou.
Pred je ak sa má najprv vykonať plus/mínus, za je až po vykonaní príkazu, v ktorom je.

```
#include <conio.h>
#include "kor.cpp"
void main () {
Kor k;
 int i;
 for (i=1; i<=4; i++) { // nasledujuce prikazy sa 4x zopakuju
cout << "zapisem" endl;
 getch (); }
```

príkaz i++ znamená to isté ako i=i+1 ... obsah premennej i sa zvýš

Príklad 8
Príklad 9
Príklad 10

V C++ je:
logická nepravda reprezentovaná celým číslom = 0,
logická pravda reprezentovaná celým číslom rôznym od 0,

Môžeme porovnávame hodnoty dvoch výrazov pomocou relačných operátorov:

výsledkom porovnania je logická pravda (celé číslo = 1) alebo logická nepravda (číslo = 0),
dva výrazy a, b môžeme porovnať pomocou nasledujúcich relačných operátorov:

matematicky:	zápis v C++:	príklad:	výsledok:
a = b	*a==b*	5==5	1
a je rôzne od b	*a!=b*	5!=5	0
a menšie alebo rovné b	*a<=b*	4<=7	0

Môžem používať nasledujúce logické operátory:

matematicky:	v C++:	príklad:	výsledok:
logické NOT a	*!a*	!0	1
logické a AND b	*a && b*	1 && 0	0
logické a OR b	*a || b*	1 || 0	1

Funkcie v C++e

existujú aj matematické funkcie - umiestnené v matematickej knižnici math.h

abs	absolútna hodnota
labs	absolútna hodnota typu long
fabs	absolútna hodnota typu float
div	vráti výsledok spolu so zvyškom delenia
ldiv	vráti výsledok spolu so zvyškom delenia typu long
sqrt	druhá odmocnina
pow	druhá mocnina
log	prirodzený logaritmus
log10	logaritmus so základom 10
sin	sínus
cos	kosínus
tan	tangens
ceil	zaokrúhli nahor
floor	zaokrúhli nadol
rand()	vylosuje náhodné číslo medzi 0 a 1
rand() %32	vylosuje náhodné číslo medzi 0 a 32

Priraďovacie:

i += j; (i = i + j)
i -= j; (i = i – j)
i *= j; (i = i * j)
i /= j; (i = i / j)
i %= j; (i = i % j)

Zaokruhlenie

```
double cislo1;
cin >> cislo1;
int zaokrouhlenecislo = cislo1 + 0.5;
```

Delenie a zvyšky

x = 22 div 3; premenná x nadobudne hodnotu 7 (22/3=7,...)
y = 22 % 3; premenná y nadobudne hodnotu 1 (zvyšok po delení 22/3 je 1)

!!! Na typ premennej

(int) 9 / 2 = 4 (ako vidíte, nie je to presné)
(float) 9 / 2 = 4.500000 (6 desatinných miest)

Príklad 11
Príklad 12
Príklad 13

Zaokrúhlenie

floor(3,99); vráti len celočíselnú časť =3, vlastne odreže desatinné miesta, výsledok je integer
ceil(3,99); je 4

Mocniny

pow(4); vráti druhú mocninu 4 = 16
sqrt(4); vráti druhú odmocninu 4 = 2

Príklad 14

Vytvorenie vlastnej fukcie

V programe rozlišujeme dve miesta:

Miesto, kde funkciu definujeme.
Miesto, kde funkciu použijeme - voláme ... vidíme, že funkciu voláme na troch miestach.

Funkciu musíme definovať skôr, ako ju použijeme.

Zadefinovanie funkcie spočíva v napísaní hlavičky funkcie a naprogramovaní jej tela:

hlavička funkcie:

hlavička funkcie začína vyhradeným slovom napr. float - tým sme povedali, že výsledkom funkcie bude desatinné čís

ďalej nasleduje meno funkcie je f,

za menom funkcie nasledujú obyčajné zátvorky, v ktorých sme uviedli typ a meno parametra,

telo funkcie:

telo funkcie obsahuje príkazy v zložených zátvorkách { }, ktoré popisujú, ako funkcia počíta, ako sa vykonáva,

výsledkom funkcie bude hodnota výrazu za príkazom return.

```
int pocitaj_objem(int sirka, int vyska, int hlbka)
{
  int objem;
  objem=sirka*vyska*hloubka;
  return objem;
}
```

Parametry musí být ve stejném pořadí a stejného typu jako v deklaraci.

Volanie funkcie

```
int a=2,b=3,c=4;
cout << pocitaj_objem(a,b,c);
```

Vo funkcii môžme použiť inú funkciu.

Môžu existovať aj viac fukncií s rovnakým názvom, každá musí mať iný počet parametrov.

Grafické a zvukové príkazy v C++e

Ak chceme #include <graphics.h> treba Alt-P, Parametre a do Linker dajte:

=-lbgi
=-lgdi32
=-lcomdlg32
=-luuid
=-loleaut32
=-lole32

Projekt vyzerá takto

```
#include <cstdlib>
#include <iostream>
#include <graphics.h>
#include <math.h>
#include <conio.h>

int main( )
{
   initwindow(600, 400, "First Sample");
   srand(time(NULL));
....
}
```

Kresliace funkcie:

setbkcolor(RED);	nastaví pozadie na červeno
setcolor(RED);	nastaví pero na červenú
line(100, 150, 40,220);	nakreslí čiaru z bodu (100,150) do bodu (40,220)
circle(100, 100, 33);	nakreslí kružnicu so stredom (100,100) a polomerom 33
setfillstyle(SOLID_FILL, BLUE);	nastaví výplň na modrú
fillellipse(rand()%100,100,30,20);	nakreslí elipsu so stredom 100, 100
rectangle(100, 150, 40,220);	nakreslí obdlznik (100,150) do bodu (40,220)

Iné

delay(100);	počká 100 ms
getch ();	program čaká na stlašenie klávesy