ÁLTALÁNOS FELÉPÍTÉS

```
#include <…>

using namespace std;

globális deklarációs rész

függvényprototípusok

int main ()
{
    …
    return 0;
}

függvények
```

KÖNYVTÁRAK

`iostream` – kiíró/beolvasó műveletek
`stdlib` – rendszerüzenetek
`math` – matematikai műveletek
`string` – szövegműveletek
`fstream` – fájlkezelés
`iomanip` – sormanipuláció

VÁLTOZÓK

`int, long:`	egész szám
`float, double:`	valós szám
`bool:`	logikai
`string:`	szöveg (karakterlánc)
`char:`	karakter
`int szamok[10]:`	10 elemű egészekből álló tömb, a tömbindexelés 0-val kezdődik

Konstans

```
const típus azonosító=érték
```

Rekord

globális részben

```
typedef struct név
{
  mezőtípus mezőazonosító;
  …
}
```

CIKLUSOK

- ***számláló ciklus***

```
for (int i=1;i<=n;++i)
{
   utasítások …
}
```

- ***előltesztető ciklus***

```
while (feltételek)
{
   utasítások …
}
```

- ***hátultesztelő ciklus***

```
do
{
   utasítások …
} while (feltételek);
```

ELÁGAZÁSOK

- ***Kétirányú elágazás***

```
if (feltételek)
{
   …
}
else if (feltételek)
{
   …
}
```

- ***Többirányú elágazás***

```
switch (kifejezés)
{
   case érték1: utasítás(ok); break;
   …
   default: utasítás(ok); break;
}
```

ELJÁRÁS, FÜGGVÉNY

```
void eljárásnév(paraméter típus paraméter azonosító, …)
{
  …
}

típus függvénynév (paraméterek)
{
  …
return visszatérési érték;
}
```

referenciális (*cím szerinti*) paraméterátadás:
`paramétertípus &paraméterazonosító`

OPERÁTOROK

`=`	értékadás
`==`	egyenlőség vizsgálat
`v+=a`	v=v+a
`i++`	i=i+1 (*inkrementáció*)
`&&`	és művelet
`||`	vagy művelet
`!`	tagadás (negáció)

SZÖVEGMŰVELETEK

`s.length():`	s hossza
`s.at(i):`	s i-edik karaktere
`s.find(minta):`	minta helye s-ben
`toupper(s):`	nagybetűsítés
`s.replace(tól,db,mivel):`	szövegcsere
`getline(cin,s,'\n'):`	beolvasás sorvégig
`setw(szám)`	oszlopos elrendezés, *pl.: cout << setw(5) << változó*

EGYÉB

`cout << „szöveg" <<`	változó kiírás a standard outputra
`cin >> változó`	beolvasás a standard inputról
`%`	modulo osztás
`//`	egysoros megjegyzés
`/* … */`	többsoros megjegyzés
`system(„pause")`	várakozás billentyűleütésre
`system(„cls")`	képernyőtörlés
`'/n'`	sorvég jel (*enter*)

FÁJLMŰVELETEK

Szükséges unit: **fstream**
Szükséges változótípusok: **ifstream**, **ofstream**

`f.open(nev.c_str())`	f fájl megnyitása
`f.close()`	f bezárása
`getline(f, string)`	sor beolvasása
`f >> s`	beolvasás s változóba
`f << s`	írás f fájlba

ELLENŐRZÖTT ADATBEOLVASÁS

```
bool hiba;
do
{
  cout << „üzenet…”;
  cin >> változó;
  hiba=cin.fail() || cin.peek()!='\n' || egyéb feltételek…;
  //a sorvégjelet és a típust ellenőrző függvények
  if (hiba)
  {
    cout << „hibaüzenet…”;
    cin.clear(); // hiba törlése
    cin.ignore(100,'\n'); //puffer ürítése
  }
} while (hiba);
```

VEKTOR

Szükséges unit: **vector**

`vector <type> nev`	vektor deklarálása
`vector <type> nev(n)`	n elemű vektor deklarálása
`bool b`	
`b=nev.empty()`	a vektor üres –e
`nev.size()`	a vektor mérete
`nev[i]`	a vektor i-edik eleme
`nev.push_back(e)`	e elem hozzáfűzése a vektorhoz
`nev.pop_back()`	elhagyja az utolsó elemet
`nev.front/back()`	megadja a vektor első/utolsó elemét
`nev.clear()`	törli a vektor minden elemét
`vec.resize(n,e)`	átméretezés, felülírás
`nev=nev2`	másolás

MODULOK

```
nev.h tartalma
#ifndef POLINOM_H_INCLUDED //include guard
#define POLINOM_H_INCLUDED
…
#endif
```

OSZTÁLYOK

```
class nev {
  private:
    változók, függvények…
  public:
    nev(): változó(érték), … {} //konstruktor
    változók, függvéynek…
    ~nev() {} //destruktor
}
```

PÉDÁNYOSÍTÁS

`Type var(konstruktor paraméterei);` példányosítás

`Type* var = new Type(konstruktor paraméterei);` példányosítás, az objektum a heap-en jön létre.

OPERÁTOR TÚLTERHELÉSE

```
class nev {
     …
     void operator =(param) {…}
}
```

ÖRÖKLŐDÉS

```
class subclass : public baseclass {…}
```

- Ha a szülőosztály konstruktorának van paramétere, akkor azt megfelelő formában meg kell hívni:
  `subclass(int param): baseclass(param) {…}`
- Metódusok felüldefiniálásához a szülőosztályban virtual kulcsszóval kell ellátni a metódust, tisztán virtuális absztrakt metódus esetén az osztály nem példányosítható:
  `virtual method(param) = 0 // pure virtual`

KIVÉTELKEZELÉS

```
enum ExceptionType(ex, …)
```
kivételtípus

```
throw ex
```
kivétel „eldobása”

```
try {
```
kivételkezelés

```
} catch (ExceptionType ex) {
if (ex == ex_1) {…}
}
```