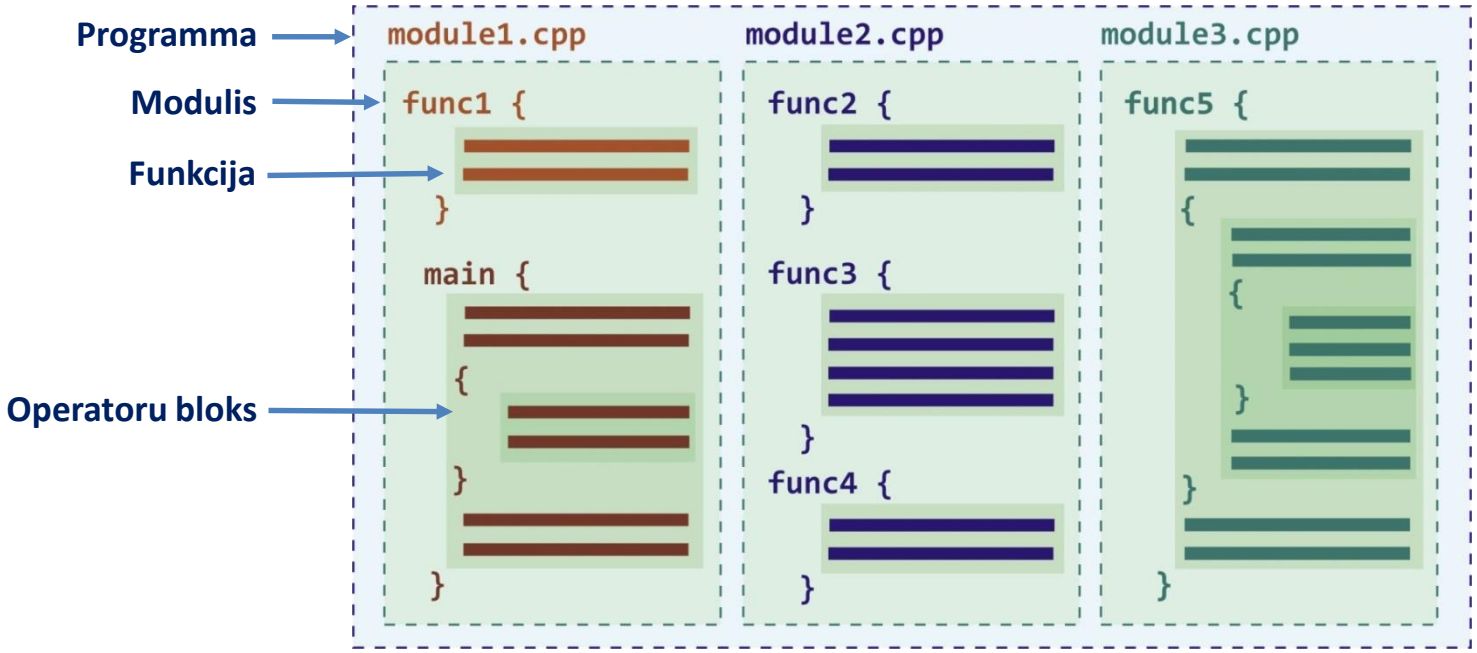
Programmētāju skola
3. līmeņa grupa

glabāšanas klases

Igors Ščukins
RTU Daugavpils

Mainīgo glabāšanas klases

Katram mainīgajam kompilators piešķir **glabāšanas klasi**, kura noteic kad mainīgais būs izveidots atmiņā un atbrīvots no atmiņas un no kuras programmas daļas šis mainīgais būs pieejams.



Dzīves laiks – tas ir laika intervāls, starp mainīgā izveidošanas un dzēšanas momentiem. Dzīves laikam var būt viena no divām vērtībām: programmas darba laiks vai operatoru bloka darba laiks.

Darbības apgabals – tas ir programmas daļa, kurā mainīgais ir "redzams" t.i. var būt izmantots. Darbības apgabals var būt programma, modulis vai operatoru bloks.

Automātiskā glabāšanas klase

Automātiskā glabāšanas klase tiek piešķirta lokālajiem mainīgajiem pēc noklusējuma. Tādi mainīgie izveidojas automātiski operatoru bloka sakumā un atbrīvojas automātiski operatoru bloka beigās.

T.i. automātisko mainīgo dzīves laiks un darbības apgabals ir operatoru bloks.

```
int main()
{
    ...
    {                   // dzīves laika sākums
        int var = 0;    //   automātiskais mainīgais
        ...             //   darbības apgabals
    }                   // dzīves laika beigas
    var = 1;            // Error: undeclared identifier
    ...
}
```

Ārējā glabāšanas klase

Ārējā glabāšanas klase tiek piešķirta visiem globālajiem mainīgajiem pēc noklusējuma. Tādi mainīgie var būt izmantoti jebkurā programmas vietā, tai skaitā visos moduļos.

T.i. ārējam globālajam mainīgajam dzīves laiks un darbības apgabals ir programma.

Tā kā kompilators kompilē katru moduļi atsevišķi, lai piekļūtu ārējos mainīgos no moduļos, kur tie nav deklarēti, nepieciešams atkārtot mainīgā deklarāciju ar modifikatoru **extern**.

module1.cpp

```
int var = 0;

void incVar() {
    var++;
}

void showVar() {
    cout << "var = " << var << "\n";
}
```

module2.cpp

```
extern int var;
extern void incVar();
extern void showVar();

void getVar() {
    cout << "var = ";
    cin >> var;
}

int main()
{
    getVar();
    incVar();
    showVar();
}
```

Moduļu mijiedarbība

Lai organizētu programmas moduļu mijiedarbību tradicionāli katram modulim izveido virsraksta failu ar paplašinājumu **.h**., kurā atkārto ārējos mainīgo un funkciju deklarāciju ar modifikatoriem **extern**. Tad lai piekļūtu moduļa mainīgajiem un funkcijām citos moduļos, var vienkārši pieslēgt moduļa virsraksta failu ar direktīvu **#include**.

module1.cpp

```
int var = 0;

void incVar() {
    var++;
}

void showVar() {
     cout << "var = " << var << "\n";
}
```

module1.h

```
extern int var;
extern void incVar();
extern void showVar();
```

module2.cpp

```
#include "module1.h"

void getVar() {
    cout << "var = ";
    cin >> var;
}

int main()
{
    getVar();
    incVar();
    showVar();
}
```

Neatkarīgā kompilēšana

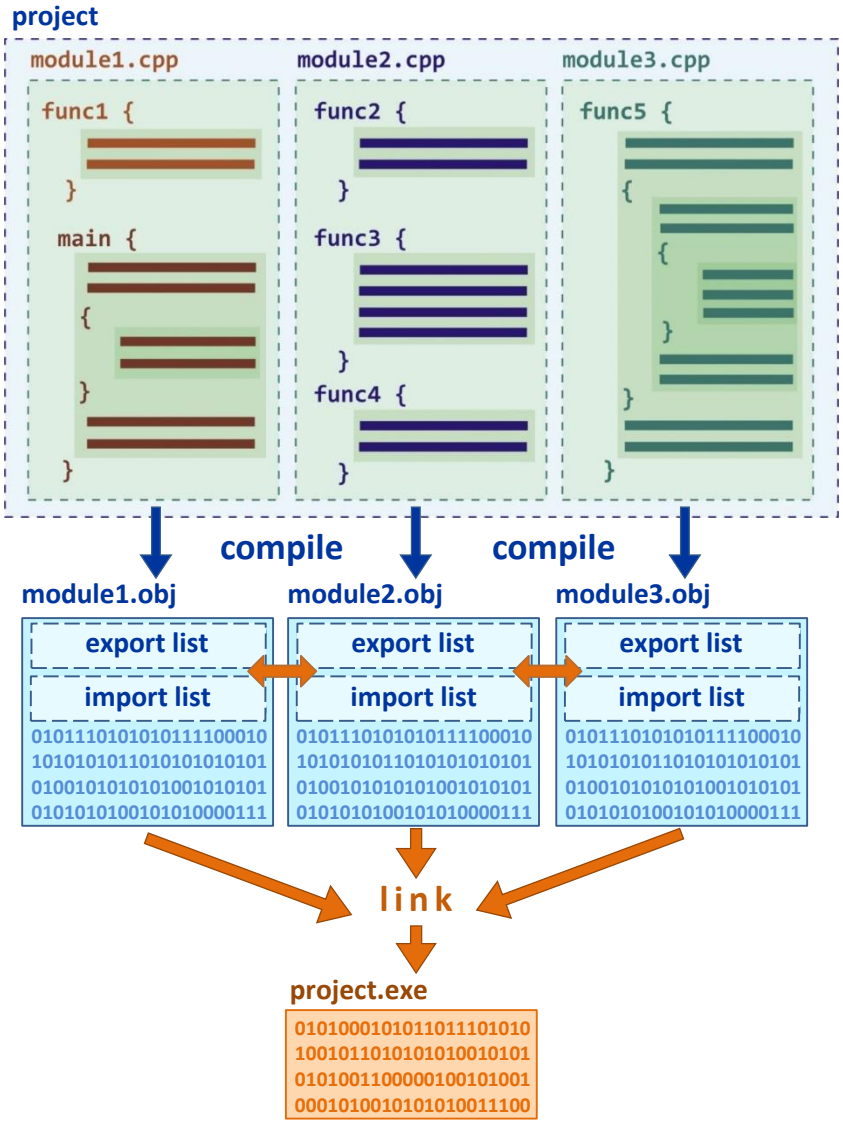
Lai atvieglotu programmu izstrādi un paātrinātu kompilāciju, lielus projektus sadala moduļos.

Katrs modulis eksportē savus ārējus identifikatorus (funkcijas un mainīgie) koplietošanai un var importēt identifikatorus no citiem moduļiem.

Kompilators kompilē katru moduli atsevišķi un izveido objekta failu (obj) no pirmkoda faila (cpp).

Katram objekta failam kompilators pievieno eksportēšanas sarakstu, kurā norāda moduļa ārējos identifikatorus, un importēšanas sarakstu, kurā norāda identifikatorus, kas tiek izmantoti, bet nav īstenoti modulī.

Saišu redaktors (linker) apvieno objektu failus izpildāmā failā (exe), saskaņojot šos importēšanas un eksportēšanas sarakstus.



Ārējā statiskā glabāšanas klase

Ārējie statiskie mainīgie – tie ir globālie mainīgie, kuri var būt izmantoti tikai tajā modulī, kur tie ir deklarēti. Tādējādi dzīves laiks tādiem mainīgajiem paliks visa programma, bet darbības apgabals ir ierobežots ar moduli. Statisko mainīgo deklarācijas laikā pievieno modifikatoru **`static`**.

module1.cpp

```
static int var = 0;

void incVar() {
    var++;
}

void showVar() {
     cout << "var = " << var << "\n";
}

void getVar() {
    cout << "var = ";
    cin >> var;
}
```

module1.h

```
extern void incVar();
extern void showVar();
extern void getVar();
```

module2.cpp

```
#include "module1.h"

int main()
{
    getVar();
    incVar();
    showVar();
}
```

Iekšējā statiskā glabāšanas klase

Iekšējie statiskie mainīgie – tie ir lokālie mainīgie, kuru deklarācijas laikā ir norādīts modifikators **static**. Darbības apgabals tādiem mainīgajiem paliks operatoru bloks, bet dzīves laiks ir palielināts līdz visai programmai. T.i. statiskā mainīgā vērtība netiks pazaudēta pēc operatoru bloka pabeigšanas.

```
void step() {
    int var = 0;
    cout << "var = " << var++ << "\n";
}

int main()
{
    for (int i = 0; i < 5; i++) {
        step();
    }
}
```

```
0 0 0 0 0
```

```
void step() {
    static int var = 0;
    cout << "var = " << var++ << "\n";
}

int main()
{
    for (int i = 0; i < 5; i++) {
        step();
    }
}
```

```
0 1 2 3 4
```