

11. lekce

Dělení programů na soubory

Dělení programu na soubory

- Je vhodné pro přehlednost.
- Umožňuje samostatnou práci s jednotlivými soubory, to umožňuje snadněji týmovou práci na projektu.
- Samostatné „hotové“ soubory lze „nainkludovat“ i do jiných projektů.
- Pro každý objekt se obvykle vytváří dva soubory. První soubor (hlavičkový) obsahuje definici atributů, metod a jejich rozhraní (definice datového typu metody a její parametry)
 - soubor má koncovku *.h. Druhý soubor obsahuje implementaci metod (kód) – soubor má koncovku *.cpp.
- Při „inkludování“ vlastních souborů musí být za ***include*** relativní adresa souboru v uvozovkách
např.: ***#include “relativní_adresa“***
Relativní adresa je zbytek cesty z výchozí složky: *složka/soubor.cpp*, pokud je *soubor.cpp* ve stejné složce jako *main.cpp*, pak je adresa pouze *soubor.cpp*

- V případě, že implementujeme metody objektu mimo jeho jmenný prostor, pak musí být u každé funkce jmenný prostor definován:

```
class student
{
public:
    student(string jmeno, string prijmeni);
private:
    string m_jmeno;
    string m_prijmeni;
};

student::student(string jmeno, string prijmeni):m_jmeno(jmeno),
                                                m_prijmeni(prijmeni)
{
}
```

Konstruktor je implementovaný mimo objekt student.

- V souborech, kde jsme definovali objekty, není vhodné používat *using namespace std*. Jestliže v některých projektech nechceme používat jmenný prostor (seznam příkazů) std, nepoužijeme v odděleném souboru ***using namespace std***. To pro nás znamená, že před každou funkcí ze jmenného prostoru std musíme jmenný prostor definovat (použijí se dvě dvojtečky).

Např.: ***std::cout<<.....<<std::endl;***

- Do jmenného prostoru std patří hlavně:
 - string
 - všechny datové kontejnery z knihovny stl
 - všechny funkce iostream
 - matematické funkce,...

Příklad editace projektu (soubory Datup.cpp a Datum.h):

– hlavičkový soubor *datum.h*

```
#ifndef Datum_846519865346533516516453135           //jedinečné označení konstanty v rámci operačního systému
#define Datum_846519865346533516516453135
#include <iostream>
#include <iomanip>
class Datum
{
private:
    unsigned m_den;
    unsigned m_mes;
    unsigned m_rok;
public:
    Datum (unsigned den, unsigned mesic, unsigned rok);
    friend std::ostream & operator << (std::ostream & o, const Datum & x);    //friend funkce ostream operator <<
};
#endif
```

#ifndef a *#endif* – slouží pro kompilátor, aby neinkludoval stejný soubor vícekrát (datum je inkludované ve více souborech projektu, takto zajistíme, že se do vytvořeného projektu připojí jenom jednou. Kdybychom to takto nevyřešili, kompilace skončí chybou.

Kompilátor kompiluje sekci mezi *ifndef* a *endif* pouze v případě, že není definována konstanta uvedená za *ifndef* (konstanta je definována v *define*).

Při kompilaci po prvním *include* se kompilace provede, při další potřebě kompilace je zjištěna existence konstanty a podruhé se tato část nekompiluje.

– implementace *datum.cpp*

```
#include "datum.h"
//-----
Datum::Datum (unsigned den, unsigned mesic, unsigned rok) : m_den(den), m_mes(mesic), m_rok(rok)
{
    if ((mesic < 1) || (mesic > 12) || (den < 1)) throw "Spatne Datum";
    if (mesic == 2)
    {
        if (den > 29) throw "Spatne Datum";
    }
    else if ((mesic == 1) || (mesic == 3) || (mesic == 5) || (mesic == 7) || (mesic == 8) || (mesic == 10) || (mesic == 12))
    {
        if (den > 31) throw "Spatne Datum";
    }
    else if (den > 30) throw "Spatne Datum";
}
//-----
std::ostream & operator << (std::ostream & o, const Datum & x)
{
    return o << std::setfill(' ') << std::setw(2) << x.m_den << "."
        << std::setfill(' ') << std::setw(2) << x.m_mes << "."
        << x.m_rok;
}
//-----
```

Datum::Datum.... - konstruktor Datum ze jmenného prostoru Datum

std::setfill(' ') << std::setw(2) – setfill definuje znak, který bude následující výpis doplněn na požadovanou šířku, setw definuje požadovanou šířku následujícího výpisu (zarovnání dd a mm)

Kompilace projektu

V případě, že jsou všechny soubory ve stejné složce, pak se projekt kompiluje příkazem:

```
g++ -std=c++11 -Wall -pedantic *.cpp
```

V případě, že některé soubory ve stejné složce nejsou, pak místo *.cpp budou uvedeny relativní cesty ke všem souborům oddělené mezerami:

win: g++ -std=c++11 -Wall -pedantic main.cpp datum.cpp clovek.cpp slozka\trida.cpp

linux: g++ -std=c++11 -Wall -pedantic main.cpp datum.cpp clovek.cpp slozka/trida.cpp

Pro potřeby ukázky v Linux Mint:

– instalace g++ (v terminálu):

su

apt-get install gcc

apt-get install build-essential

v případě chyby instalace se spustí řádek:

apt-get update

v terminálu provedeme kompilaci pomocí příkazu

make

ten spustí soubor **Makefile** (nutno dodržet velké a malá písmena).

Makefile je kompilační soubor, který jsme sami vytvořili pro potřeby kompilace. Nemá příponu!

Příklad kompilačního souboru Makefile z ukázkového příkladu (# uvozuje komentář):

Pozor, mezery mezi řádky a tabulátory jsou povinné!

CXX=g++ *#kompilator*

LD=g++ *#linker (kompilátor nejprve zkompiluje jednotlivé soubory a linker je spojí)*

CXXFLAGS=-std=c++11 -Wall -pedantic *#prepinace kompilatoru*

all: program *#pred : je to, co ma udelat, za : je to, co je k tomu potreba a na radku pod tim je, jak se to ma udelat*

program: main.o datum.o clovek.o trida.o *# program: jmeno spustitelneho souboru, bez koncovky*
 \$(LD) \$(CXXFLAGS) -o \$@ \$^ *# \$()-hodnota promenne, \$@ je vsechno pred :, \$^ je vsechno za :*

main.o: main.cpp trida.h clovek.h datum.h
 \$(CXX) \$(CXXFLAGS) -c -o \$@ \$< *# \$< je prvni string za :, -c je compile bez vytvoreni spustitelneho souboru*

datum.o: datum.cpp datum.h
 \$(CXX) \$(CXXFLAGS) -c -o \$@ \$< *# -o je vytvoreni objekt file, to je jeden zkompilovany soubor, ze ktereho linker*
 # vytvori spustitelny soubor

clovek.o: clovek.cpp clovek.h datum.h
 \$(CXX) \$(CXXFLAGS) -c -o \$@ \$<

trida.o: trida.cpp trida.h clovek.h datum.h
 \$(CXX) \$(CXXFLAGS) -c -o \$@ \$<

run: program
 ./program *#spusteni programu po prikayu make run*

clean:
 rm -f *.o *#smaze prechodne objekt files*

clear:
 rm -f *.o *#smaze prechodne objekt files (pouzije se clean nebo clear)*