

Programovacie techniky

6. string, fstream, Abstract data type, Standard template library (STL), stack, vector, map, list, pair, iterátor, try/catch (exception handling), heap

Konstruktor/Deštruktor

Po zavolaní vytvorí objekt:

List();

Deštruktor – čo spraviť po zavolaní delete:

~List();

Deštruktor nemá parameter
delete neumožňuje poslať parameter

delete this

```
class Test {  
    public:  
        int a, b;  
        void kill(){  
            delete this; //OK alebo nie?  
            this->a = 10; //objekt už neexistuje  
        }  
};
```

delete this je zlý nápad

string

```
#include <iostream>
#include <string>
using namespace std;

int main () {
    string str1 = "Hello", str2 = "World", str3;
    int len ;

    str3 = str1; //prekopíruj str1 do str3
    cout << "str3 : " << str3 << endl;

    str3 = str1 + str2; //spojit' str1 a str2
    cout << "str1 + str2 : " << str3 << endl;

    len = str3.size(); //vel'kost' str3
    cout << "str3.size() : " << len << endl;

    return 0; }
```

string

```
#include <string>
using namespace std;

int main () {
    string str1 = "remove aaa";
    str1.erase(7, 3); //zmaže aaa

    string str2 = "ade";
    str2.insert(1, "bc"); //abcde

    return 0;
}
```

string

```
#include <iostream>
#include <string>

int main () {
    std::string str="We think in generalities, but we live in
    details.";

    std::string str2 = str.substr (12,12);    // "generalities"

    unsigned pos = str.find("live");          // pozícia "live" v
    str (33). Prvý výskyt reťazca.

    std::string str3 = str.substr (pos);      //od "live" až po
    koniec

    return 0;
}
```

string

```
#include <iostream>
#include <string>
using namespace std;

int main () {
    string str="password";

    if(str == "password")
        cout << "ur using a weak password";

    return 0;
}
```

string: c_str()

```
#include <string>
#include <cstring> //string.h, C knižnica
using namespace std;

int main () {
    string str1 = "I love PT!";
    char str[100];
    strcpy (str, str1.c_str());
    return 0;
}
```

c_str() potrebné pre spätnú kompatibilitu s C
c_str() vráti const char*

string: c_str()

```
#include <string>
```

```
#include <cstdio>
```

```
using namespace std;
```

```
int main () {
```

```
    string str1 = "1 8 16 23 78";
```

```
    int a1, a2, a3, a4, a5;
```

```
    sscanf(str1.c_str(), „%d %d %d %d %d“,  
&a1, &a2, &a3, &a4, &a5);
```

```
    return 0;
```

```
}
```

fstream: práca so súbormi

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main () {
    ofstream subor;
    subor.open ("example.txt");
    subor << "I love PT!\n";
    subor.close();
    return 0;
}
```

fstream: práca so súbormi

```
#include <iostream>
#include <fstream>
using namespace std;

int main () {
    ofstream subor ("example.txt");
    if (subor.is_open()) {
        subor << "I love PT!\n";
        subor.close(); //void close()
    }
    else cout << "Subor nebolo mozne otvorit!";
    return 0;
}
```


ofstream

```
ofstream subor ("example.txt"); //zápis, nie binary
```

```
ofstream subor ("example.txt", ios::app); //pridaj k  
existujúcemu súboru
```

```
ofstream subor ("example.txt", ios::binary);  
//binárny zápis
```

```
ofstream subor ("example.txt", ios::app |  
ios::binary); //kombinácia
```



```
ios::out //default pre ofstream
```

fstream: čítanie

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
    string line;
    ifstream subor ("example.txt");
    if (subor.is_open()) {
        while ( getline (subor, line) ) { //načítaj riadok
            cout << line << '\n';
        }
        ...
        return 0;
    }
}
```

ifstream

```
ifstream subor ("example.txt"); //čítanie, nie  
binary
```

```
ifstream subor ("example.txt", ios::binary);  
//binárne čítanie
```

```
ios::in //default pre ifstream
```

bool eof()

```
ifstream subor ("example.txt"); //čítanie, nie  
binary
```

```
if(subor.eof()) { //koniec súboru  
    ...  
}
```

Abstract data type (ADT)

Matematický model pre dátové štruktúry

K dátam možno pristúpiť a manipulovať s nimi len cez malú množinu definovaných funkcií

Rôzne implementácie – funkčnosť rovnaká, ale rôzna rýchlosť

Štruktúra dát a manipulácia s nimi môže byť rôzna, ale výsledok operácií musí byť rovnaký

Pomocou ADT možno rýchlejšie pochopiť veľké programy

ADT: príklady

Stack (zásobník)

Queue (fronta,rad)

Deque (double-ended queue, obojsmerná fronta)

List (zoznam)

Set (množina)

Priority queue (utriedený zoznam)

Map (zobrazenie)

Vector (dynamické pole)

STL: standard template library

STL obsahuje implementácie ADT

ADT: príklady

Fronta – ADT typu FILO (first in, last out)

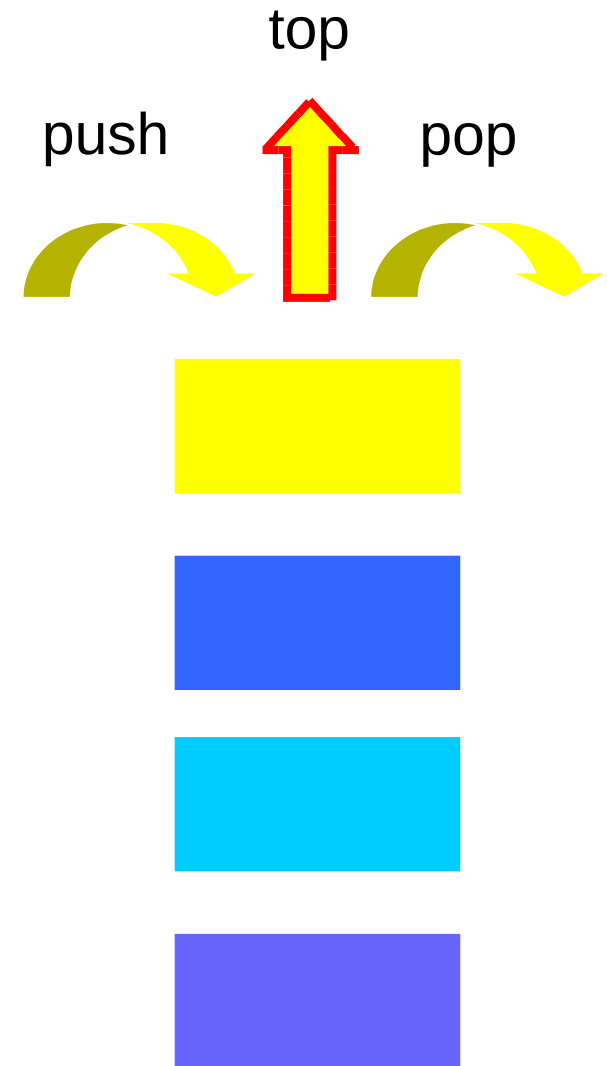
operácie:

push(prvok) :vlož prvok

pop() : vyber posledne pridaný
prvok

empty() : vráti true, keď je
zásobník prázdny

top() :ukáž posledne pridaný
prvok



Kontajner (container)

Container je trieda pomocou ktorej je implementovaný ADT

Stack môže byť implementovaný ako:

vector, deque and list

ADT musí v tomto prípade podporovať metódy:

back (top), push_back, pop_back

Stack: metódy

`empty()`: je stack prázdny?

`size()`: vráť veľkosť

`top()`: posledne vložený prvok

`push()`: vlož prvok

`pop()`: odstráň prvok

Stack: metody (C++11)

`emplace()`: vytvoř a vlož prvek

`swap()`: vymeň obsah dvou stackov

Stack: příklad

```
#include <iostream>
```

```
#include <stack>
```

```
int main () {  
    std::stack<int> mystack;  
    for (int i=0; i<5; ++i) mystack.push(i);  
  
    std::cout << "size of mystack: " <<  
mystack.size() << '\n';  
    return 0;  
}
```

Vector

size

max_size: max. povolená veľkosť

empty

push_back

pop_back

clear: odstráň všetky prvky (size=0)

resize: zmeň veľkosť

+ďalšie

Vector: **at**

```
#include <iostream>
#include <vector>

int main () {
    std::vector<int> myvector (10);
    for (unsigned i=0; i<myvector.size(); i++)
        myvector.at(i)=i;
    for (unsigned i=0; i<myvector.size(); i++)
        std::cout << ' ' << myvector.at(i);
    std::cout << '\n';

    return 0;
}
```


Vector: [], at

```
std::vector<int> myvector (10);
```

```
cislo = myvector.at(100); //out_of_range
```

```
cislo = myvector[100]; //bez overenia rozsahu
```

Pokus čítať/zapisovať mimo rozsah s **at** možné ošetriť try/catch

Vector: **at**

Be safe!

Vector: `at`

Be safe!

Pozn.: to to je C++ prednáška!

Mimo rozsah: try...catch

```
#include <iostream>
#include <stdexcept>
#include <vector>

int main () {
    std::vector<int> myvector(10);
    try {
        myvector.at(20)=100; }

    catch (std::out_of_range& oor) {
        std::cerr << "Out of Range error: " <<
        oor.what() << '\n'; }

    return 0; }
```

try...catch

```
#include <iostream>
using namespace std;

int main () {
    try {
        throw 20;
    }
    catch (int e) {
        cout << "An exception occurred. Exception Nr.
" << e << '\n';
    }
    return 0;
}
```

try...catch

```
#include <iostream>
using namespace std;

double division(int a, int
b) {
    if( b == 0 ) {
        throw "Division by
zero condition!";
    }
    return (a/b);
}
```

```
int main () {
    double z;
    try {
        z = division(50, 0);
        cout << z << endl;
    } catch (const char*
msg) {
        cerr << msg <<
endl;
    }
    return 0;
}
```

try...catch

```
#include <iostream>
```

```
#include <new>
```

```
int main () {
```

```
    try {
```

```
        int* pole = new int[10000];
```

```
    }
```

```
    catch (std::bad_alloc& ba) {
```

```
        std::cerr << "bad_alloc caught: " << ba.what()
```

```
<< '\n';
```

```
    }
```

```
    return 0;
```

```
}
```

map

```
#include <map>
#include <iostream>
using namespace std;

int main () {
    map <string, int> hodnotenie;

    hodnotenie["Martin"] = 1;
    hodnotenie["Imro"] = 4;
    hodnotenie["Walter"] = 5;

    cout << hodnotenie["Imro"] << endl; //4
    return 0;
}
```


Iterátory: begin, end

```
#include <vector>
#include <iostream>
using namespace std;

int main() {
    vector<int> v; vector<int>::iterator itv;
    v.push_back(22); v.push_back(22997845);

    itv = v.begin();
    while(itv != v.end()) {
        cout << *itv << ' ';
        itv++;
    }
}
```

Iterátory: begin, end

```
std::map<char,int>::iterator zac=mymap.begin();
```

```
std::map<char,int>::iterator konec=mymap.end();
```

Potrebné na prechádzanie prvkami map, vector, stack, string...

Iterátory (iterators)

```
#include <iostream>
#include <map>

int main () {
    std::map<char,int> mymap;
    std::map<char,int>::iterator it;

    mymap['b'] = 100; mymap['a'] = 200;
    mymap['c'] = 300;

    for (it=mymap.begin(); it!=mymap.end(); ++it)
        std::cout << it->first << " ==> " << it->second <<
'\n';
}
```

pair: first, second

```
#include <utility>    //std::pair, std::make_pair
#include <string>
#include <iostream>
```

```
int main () {
    std::pair <int,double> p1;
    p1=std::make_pair(1, 0.99);

    std::cout << p1.first << "\t" << p1.second <<
std::endl;

    return 0;
}
```

list

```
#include<list> //obojsmerne zret'azený zoznam
using namespace std;
int main() {
    list<int> a;
    a.push_back(1);    // 1
    a.push_front(2);   // 2 -> 1
    a.push_front(3);   // 3 -> 2 -> 1
    a.size();          //veľkosť je 3
    a.pop_front();     // 2 ->1
    a.pop_back();      // 2
    a.clear();         // vymazanie celého zoznamu
}
```

list

```
#include <list>
#include <algorithm> //sort
using namespace std;

int main () {
    double d[]={12.15, 2.72, 73.0, 12.77, 3.14,
12.77, 73.35, 72.25, 15.3, 72.25};
    list<double> l(d, d+10);

    l.sort();           //2.72, 3.14, 12.15, 12.77, 12.77,
15.3, 72.25, 72.25, 73.0, 73.35
    l.unique();         // 2.72, 3.14, 12.15, 12.77, 15.3,
72.25, 73.0, 73.35
}
```

min

```
#include <iostream>
#include <algorithm>    // std::min

int main () {
    std::cout << "min(1,2)==" << std::min(1,2) <<
'\n';
    std::cout << "min('a','z')==" << std::min('a','z') <<
'\n';

    return 0;
}
```

heap

```
#include <iostream>
#include <algorithm>
#include <vector>

int main () {
    int myints[] = {10, 20, 30, 5,15};
    std::vector<int> v(myints,myints+5);

    std::make_heap (v.begin(),v.end());
    std::cout << "initial max heap  : " << v.front() <<
'\n'; //v.front() je prvý element vo vektore
}
```


Úloha

Ako urobiť mapu zoznamov?

Alebo vektor zoznamov?

Alebo zoznam zásobníkov?

Alebo ... be happy u dont have to do it in C!

Úloha

Máme zdrojový kód (textový súbor) a máme zistiť, či v kóde sú korektne párované zátvorky { }

Príklad: {...{...}..}..} : zátvorky nie sú párované

Ako použiť Stack??

```
#include <stack>
#include <string>
using namespace std;

int main() {
    string str = "{...{...}..}..}";
    stack<string> s;

    //ako d'alej??

}
```