Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Објектно ориентирано програмирање

Аудиториски вежби 2

Верзија 1.0, 20 Февруари, 2017

Содржина

1. Вовед во C++

1.1. Вовед

- C++ вклучува многу карактеристики на C јазикот дополнети со дополнителни механизми за обкектно-ориентирано програмирање (ООП)
- Јазикот C++ е развиен во лабораториите на Bell и во почетокот бил наречен „C со класи“
- Името C вклучува во себе операција за зголемување на јазикот C () и укажува на тоа дека C++ е верзија на C со проширени можности
- C++ компајлерот може да се користи и за преведување на C програми

1.2. Влез/Излез текови во C++ `<iostream>`

Во C++ за работа со влезно/излезните текови наместо функциите `printf` и `scanf` ги користиме операторите:

- `<<` од објектот `cout` (`cout <<`)
- `>>` од објектот `cin` (`cin >>`)

Пример C

```
printf("Vnesi nova vrednost: ");
scanf("%d", &vred);
printf("Novata vrednost e: %d\n", vred);
```

Пример C++

```
cout << "Vnesi nova vrednost: ";
cin >> vred;
cout << "Novata vrednost e: " << vred << '\n';
```

При тоа препрорачливо е изолираните `\n` да се заменат со `endl`:

```
cout << "Novata vrednost e: " << vred << endl;
```

За работа со влезно/излезните текови во C++ треба да се вклучи датотеката со заглавја `<iostream>`

Пример

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

int main () {
    cout << "Vnesi gi tvoite godini : ";
    int myAge ;
    cin >> myAge ;
    cout << "Vnesi gi godinite na prijatel : ";
    int friendsAge ;
    cin >> friendsAge ;
    if (myAge > friendsAge)
        cout << "Ti se postar.\n";
    else if (myAge < friendsAge)
        cout << "Prijatelot e postar.\n";
    else
        cout << "Ti i prijatelot imate godini.\n";
    return 0;
}
```

1.3. Декларација на променливи во C++

Променливите во C++ може да се декларираат во било кој дел од програмата сè додека нивната декларација е пред употреба на променливата во некоја наредба.

```
for(int i = 0;  i < 5; i++)
    cout << i << endl;
```

Областа на делување на локалните променливи почнува од декларацијата и се протега до крајот на блокот во кој припаѓа означен со `}`. Декларација не може да се врши во делот за услов кај структурите за повторување `while`, `do/while`, `for` или кај `if`.

1.4. `inline` функции

При секој стандарден повик на функција се троши дополнително време во процесот на повикување на функцијата. Затоа во C++ при дефинирањето на мали и едноставни функции може да се употреби клучното зборче `inline` кое што наместо повик кон функцијата на местото каде што треба да се повика го вметнува самиот код на функцијата со што се избегнува дополнителното време при повикување на функцијата.

Пример функција за пресметување на волумен на коцка

```
#include <iostream>
using namespace std;
inline float cube(const float s) {
    return s * s * s;
}
int main() {
    cout << "Stranata na kockata: ";
    float strana;
    cin >> strana;
    cout << "Volumenot na kocka so strana " << strana << " e: " << cube(strana)
         << endl;
    return 0;
}
```

1.5. Дополнителни клучни зборови во C++

asm	explicit	operator
catch	friend	private
class	inline	protected
const_cast	mutable	public
delete	new	reinterpret_cast
dynamic_cast	namespace	static_cast
template	throw	using
this	try	virtual

1.6. Користење `typedef`

Користење на `typedef` е дозволено, но не и неопходно при дефинирање на структури, унии или набројувачки множества (`enum`) во C++.

Пример

```
#include <iostream>
using namespace std;
struct Person {
    char ime[80], adresa[90];
    double plata;
};
int main() {
    Person Vraboten[50]; // pole so elementi od tip struct Person
    Person Rakovoditel;
    Rakovoditel.ime = "Aleksandar";
    cout << "Imeto na rakovoditelot e" << Rakovoditel.ime << endl;
    return 0;
}
```

1.7. Функции како членови на структура

Во C++ е дозволено дефинирање на функција како дел од некоја структура.

Пример

```
struct Person {
    char ime[80], adresa[80];
    // deklaracija na funkcijata za pecatenje na imeto i adresata
    void print(void);
};
```

- Функцијата `print()` е само декларирана; кодот на функцијата се наоѓа на друго место.
- Големината на структурата (`sizeof`) е определена **само** од големината на податочните елементи. Функциите кои се деклараирани во неа не влијаат на нејзината големина. Компајлерот го имплементира ова однесување со тоа што функцијата `print()` е позната само во контекст на структурата `Person`.
- Пристапот до функција дефинирана како дел од структура е идентичен како и пристапот до податочен елемент од структура, т.е. се користи операторот `(.)`. Кога се употребува покажувач кон структура се употребува →. Оваа синтакса дозволува да се користи исто име на функција во повеќе структури.

1.8. Користење на функција како член на структура

Пример

```
#include <iostream>
using namespace std;
struct Person {
    char ime[80], adresa[90];
    double plata;
    void printperson();
};
void Person::printperson() {
    cout << "Imeto na vraboteniot e:" << ime << "a, negovata adresa e:"
            << adresa << endl;
}
int main() {
    Person Rakovoditel;
    //...fali inicijalizacija
    Rakovoditel.printperson();
    return (0);
}
```

2. Референци

2.1. Дефиниција и начин на декларација

Референца е нов податочен тип во C++ кој е сличен на покажувач во C но многу побезбеден за употреба.

Начин на декларација

```
<Type> & <Name>
```

Пример

```
int A = 5;
int& rA = &A;
```

2.2. Разлики и сличности со покажувачи

Референците во C++ се разликуваат од покажувачите во следниве разлики:

- Не може да се пристапи директно на референцата откако ќе се дефинира, секое пристапување до неа е всушност пристапување до објектот/променливата кон која референцира.
- Откако еднаш ќе се инцијализира не може да се промени да референцира кон друг објект/променлива.
- Не може да бидат NULL (да не референцираат кон ништо).
- Откако ќе се декларираат мораат веднаш да се иницијализираат.

2.3. Употреба

Една од најважните употреби на референците е при пренос на аргументи во функции.

Пример

```
int swap(int &a, int &b) {
    a += b;
    b = a - b;
    a -= b;
}
int main() {
    int x = 10;
    int y = 15;
    swap(x, y);
    // x = 15, y = 10
    return 0;
}
```

2.4. Пример аргументи референца кон структура

Пример

```
// deklaracija na struktura
struct Person {
    char ime[80], adresa[90];
    double plata;
};
Person Vraboten[50]; // poli elementi od tip struktura Person
// printperson ocekuva referenca do struktura
void printperson(Person const &p) {
    cout << "Imeto na vraboteniot e:" << p.ime << "a, negovata adresa e:"
            << p.adresa << endl;
}
// zemi gi podatocite za vraboteniot preku negoviot reden broj
Person const &ZemiVraboten(int index) {
... return (Vraboten[index]); // vrakja referenca
}
int main() {
    Person Rakovoditel;
    printperson(Rakovoditel); // nema referenca
    // promenlivata nema da se promeni vo funkcijata
    printperson(ZemiVraboten(5)); // se prenesuva referenca
    return (0);
}
```

3. Нова синтакса за кастирање

Во C се употребуваше следната синтакса за кастирање

```
(typename)expression
(float) broitel;
```

Во C++ иако е дозволена и претходната нотација, новиот начин на нотација е:

```
typename(expression)
float(broitel);
```

Во C++ се воведуваат 4 нови видови кастирање:

- `static_cast<type>(expression)` - стандардно кастирање за конверзија на еден тип во друг
- `const_cast<type>(expression)` - се користи за модифицирање на типот на константи
- `reintrepret_cast<type>(expression)` - се користи за промена на интерпретација на информацијата
- `dynamic_cast<type>(expression)` - поврзан со polymorphism

4. Пример решенија на задачите од структури во C++

Решение `oop_av24.cpp`

```
#include <iostream>
#include <cstring> // string.h math.h -> cmath
using namespace std;

struct student {
    char firstName[50];
    char lastName[50];
    int number;
    int totalPoints;
    void print() {
        cout << firstName << "\t";
        cout << lastName << "\t";
        cout << number << "\t";
        cout << totalPoints << "\n";
    }
};


void normalize(char *name, bool allUppercase = false) {
    *name = toupper(*name);
    ++name;
    while(*name) {
        if(allUppercase) {
            *name = toupper(*name);
        } else {
            *name = tolower(*name);
        }
        ++name;
    }
}

/*void normalize(char *name) {
    normalize(name, false);
}
*/
void read(student &s) {
    cin >> s.firstName;
    cin >> s.lastName;
    normalize(s.firstName);
    normalize(s.lastName);
    cin >> s.number;
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    s.totalPoints = a + b + c + d;
}



void sort(student s[], int n) {
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n - 1; ++j) {
            if(s[j].totalPoints < s[j + 1].totalPoints) {
                student tmp = s[j];
                s[j] = s[j + 1];
                s[j + 1] = tmp;
            }
        }
    }
}

int main() {
    student s[100];
    int n;
    cin >> n;
    for(int i = 0; i < n; ++i) {
```

```
        read(s[i]);
    }
    sort(s, n);
    cout << "======== ORDERED ========" << endl;
    //read(s);
    for(int i = 0; i < n; ++i) {
        s[i].print();
        //print(s[i]);
    }

    return 0;
}
```

Решение `oop_av25.cpp`

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

struct city {
    char name[100];
    int population;
};

struct country {
    char name[100];
    char president[100];
    city capital;
    int population;
    void print() {
        cout << name << "\t" << president << "\t";
        cout << capital.name << "\t";
        cout << capital.population << "\t";
        cout << population << endl;
    }
};

void print(country &c) {
    cout << c.name << "\t" << c.president << "\t";
    cout << c.capital.name << "\t";
    cout << c.capital.population << "\t";
    cout << c.population << endl;
}

void read(int n, country c[]) {
    for (int i = 0; i < n; ++i) {
        cin >> c[i].name;
        cin >> c[i].president;
        cin >> c[i].capital.name;
        cin >> c[i].capital.population;
        cin >> c[i].population;
    }
}

int sum(int a = 0, int b = 5) {
    return a + b;
}

void maxCapitalPopulation(int n, country c[]) {
    country max = c[0];
    for (int i = 1; i < n; ++i) {
        if (c[i].capital.population > max.capital.population) {
            max = c[i];
        }
    }

    cout << max.president << endl;
}

int main() {
    country countries[100];
    country &first = countries[0];
    first = countries[1];
    int n;
```

```
    cin >> n;
    read(n, countries);

    for (int i = 0; i < n; ++i) {
        //print(countries[i]);
        countries[i].print();
    }

    maxCapitalPopulation(n, countries);
/*    cout << sum(3, 10) << endl;
    cout << sum(10) << endl;
    cout << sum() << endl;*/


    return 0;
}
```

5. Изворен код од примери и задачи

https://github.com/finki-mk/OOP/

Source code ZIP