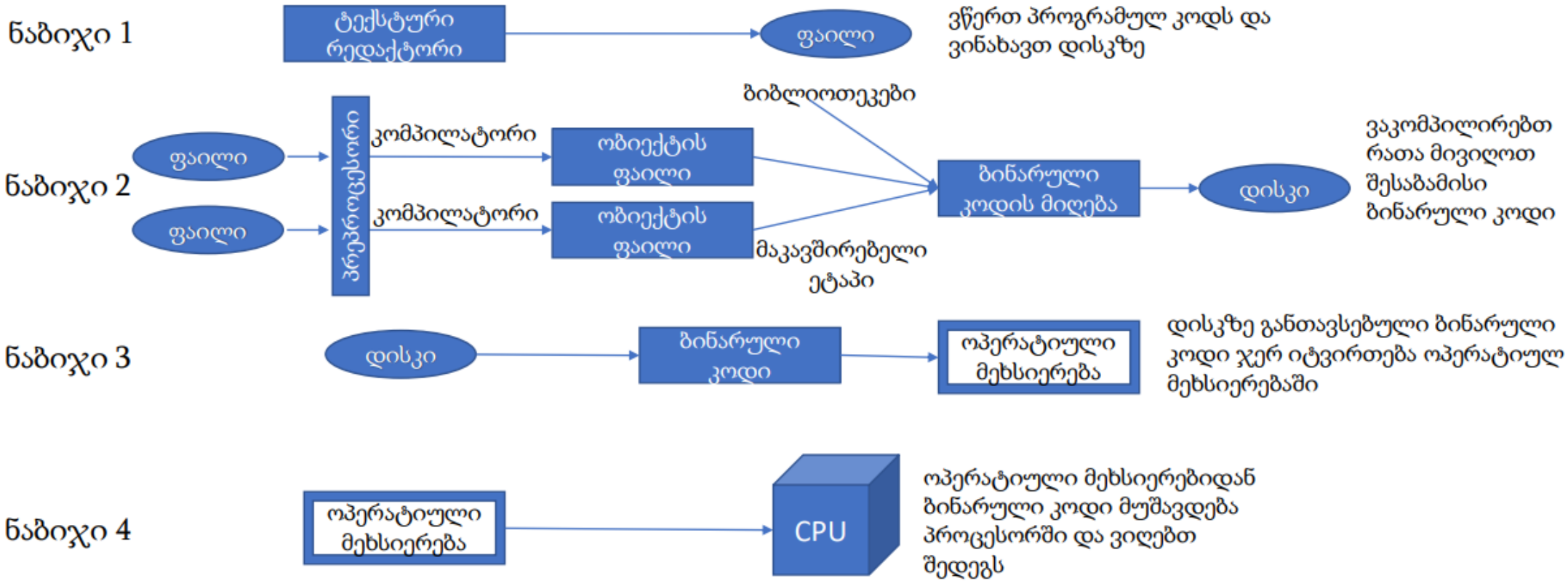
ობიექტზე ორიენტირებული დაპროგრამება

12/10/2019

ერეკლე მაღრაძე

კომპილაცია


ნაბიჯი 1
ტექსტური რედაქტორი
ფაილი
ვწერთ პროგრამულ კოდს და ვინახავთ დისკზე
ნაბიჯი 2
ფაილი
ფაილი
პრეპროცესორი
კომპილატორი
კომპილატორი
ობიექტის ფაილი
ობიექტის ფაილი
ბიბლიოთეკები
მაკავშირებელი ეტაპი
ბინარული კოდის მიღება
დისკი
ვაკომპილირებთ რათა მივიღოთ შესაბამისი ბინარული კოდი
ნაბიჯი 3
დისკი
ბინარული კოდი
ოპერატიული მეხსიერება
დისკზე განთავსებული ბინარული კოდი ჯერ იტვირთება ოპერატიულ მეხსიერებაში
ნაბიჯი 4
ოპერატიული მეხსიერება
CPU
ოპერატიული მეხსიერებიდან ბინარული კოდი მუშავდება პროცესორში და ვიღებთ შედეგს

კოდის წერა და კომპილაცია

- C++ პროგრამული კოდის დასაწერად დაგჭირდებათ ტექსტური რედაქტორი ან IDE (Integrated Development Environment)
- C++ ის კოდის საწერად ყველაზე მოხერხებული ტექსტურ რედაქტორებად შეიძლება განვიხილოთ
  atom - https://atom.io
  Visual Studio Code - https://code.visualstudio.com
- კოდის კომპიუტერზე ასამუშავებლად და გასაშვებად საჭიროა C++ (ან სხვა შესაბამისი ენის) კომპილატორი
  Linux ოპერაციულ სისტემაში - gcc https://gcc.gnu.org/
  Windows ოპერაციულ სისტემაშ - ბევრია მაგრამ შეგიძლიათ გამოიყენოთ http://www.codeblocks.org ან
  Visual Studio - https://visualstudio.microsoft.com/

C++ კოდის სტრუქტურა

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5       {
6           cout<<"Hello World"<<endl;
7           return 0;
8       }
```

- int main() - ძირითადი ფუნქცია, რომლის ტანში ჩაწერილი კოდიც სრულდება პირველ რიგში;
- int არის ფუნქციის ტიპი, ანუ იმ ცვლადის ტიპი რაც უნდა დააბრუნოს ფუნქციამ;

- #include <iostream> - მონაცემების შეტანა გამოტანის ძირითადი ბიბლიოთეკა, cin, cout;
- cin>> მონაცემების შეტანის ნაკადი
- cout<< მონაცემების გამოტანის ნაკადი
- using namespace std; - ეს არის სახელობითი სივრცე სადაც არის განსაზღვრული სტანდარტული ფუნქციები და ცვლადები, რომელიც შეიძლება დაგვჭირდეს პროგრამის წერის დროს, მაგალითად, cin, cout არის ამ სტანდარტულ (std) სივრცეშ განსაზღვრული ფუნქციები სხვა სივრცეში მათ გამოსაძახებლად საჭირო იქნება შემდეგი std::cin, std::cout;

C++ ცვლადების ტიპები

ტიპი	განმარტება
bool	ლოგიკური ცვლადის ტიპი, იღებს true, false მნიშვნელობებს
char	ერთ ბაიტიანი (8 ბიტი - 8 სიმბოლო) ცვლადის ტიპი, სადაც შეიძლება ჩაიწეროს სიმბოლო, მაგ. ‘a’, ‘1’ ანუ ჩანაწერი რასაც 8 ბიტზე მეტი არ ჭირდება
int	მთელი ტიპის რიცხვი - min (-2147483647 - 1), max (2147483647)
float	ნამდვილი ტიპის ცვლადის ტიპი, ე.წ. მცურავ წერტილიანი რიცხვითი მნიშვნელობის ცვლადის ტიპი - მძიმის შემდეგ ერთი რიცხვის სიზუსტით დამრგვალება;
double	ნამდვილი ტიპის ცვლადი, მძიმის შემდეგ ორი რიცხვის სიზუსტით დამრგვალება;
void	ტიპი, რომელიც გამოიყენება ტიპის არ ქონის აღსანიშნავად (კონკრეტულად ეს კარგად ჩანს ფუნქციების მაგალითზე);
wchar_t	char ტიპის ცვლადი, ოღონდ მეტი გამოყოფილი მეხსიერებით, მეხსიერებაში გამოყოფილი ზომა დამოკიდებულია კომპილატორზე - საჭიროა <cwchar>-ის ჩართვა პროგრამაში, include-ის გამოყენებით;

C++ სხვა ტიპები

- C++ -ში გამოიყენება კიდევ რამდენიმე ტიპი, რომელიც ნაკლებად თუ ეთანადება კონკრეტულ რიცხვით ან სიმბოლურ მნიშვნელობას და გამომდინარეობს ენაში გამოსაყენებელი მონაცემების სტრუქტურებიდან ან ფუნქციებიდან;
- Enumeration
- Pointer
- Array
- Reference
- Structure
- Class

  ასევე შესაძლებელია განსაზღვროთ თქვენთვის სასურველი ტიპი, ან მეტი - არსებულ შედგენილ ტიპს შეუცვალოთ თვისობრიობა;

ცვლადის გამოცხადება

პროგრამაში შესაძლებელია ცვლადის გამოცხადება

type variable_name = value;

extern int d = 3, f = 5; //d და f ცვლადების გამოცხადება;
int d=3, f=5; // d და f განსაზღვრა და ინიციალიზაცია;
byte z=22; // z განსაზღვრა და ინიციალიზაცია;
char x='x'; // x განსაზღვრა და ინიციალიზაცია;

extern

- extern - გამოიყენება ცვლადის გამოცხადებისთვის პროგრამის ნებისმიერ კომპონენტში;
- ცვლადის გამოცხადება კომპილატორს უზრუნველყოს ინფორმაციით რომ მოცემული ტიპის ერთი ცვლადი მაინც არსებობს - ანუ ერთი ცვლადისთვის მაინც არის გამოყოფილი მეხსიერება;
- extern - ძალიან ეფექტურად გამოიყენება, როდესაც კოდი განაწილებულია რამდენიმე ფაილში. ამ ბრძანების საშუალებით ყველა ცვლადის და ტიპის გამოცხადება შეგვიძლია მოვათავსოთ ერთ ფაილში, რაც ამარტივებს კოდის წერის პროცესს და მეტად კითხვადს და გასაგებს ხდის კოდს;

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

//ცვლადების გამოცხადება:
extern int a,b;
extern int c;
extern float f;

int main()
    {
        //ცვლადების განსაზღვრა
        int a,b;
        int c;
        float f;

        //ინიციალიზაცია
        a = 10;
        b = 20;
        c = a + b;

        cout << c << endl;

        f = 70.0/3.0;
        cout << f << endl;

        return 0;
    }
```

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

int func();

int main()
    {
        int i = func();
        return 0;
    }

int func()
    {
        return 10;
    }
```

შესაძლებელია ფუნქციის გამოცხადება და განსაზღვრა, ფუნქციის ინიციალიზაცია არ ხდება;

lvalue, rvalue

- lvalue - გამოსახულება, რომელიც აღწერს მეხსიერების გარკვეულ მონაკვეთს ეწოდება lvalue.
- lvalue - შეიძლება გამოვიყენოთ გამოსახულების, როგორც მარცხენა ისე მარჯვენა მხარეს, მაგ:
  int a = 20;
  int b = 30;
  a = b;
- rvalue – არის გამოსახულება შეესაბამება იმ მნიშვნელობას (მონაცემს) რომელიც წერია მეხსიერების შესაბამის მონაკვეთში;
- rvalue - შეიძლება გამოვიყენოთ გამოსახულების მხოლოდ მარჯვენა მხარეს, მაგ:
  20 = 30; დაუშვებელი გამოსახულებაა

ლოკალური ცვლადები

- ნებისმიერი ცვლადი, რომელიც გამოცხადებულია მხოლოდ კონკრეტული ფუნქციის შიგნით, არის ***ლოკალური ცვლადი*** - ამ ფუნქციის ლოკალური ცვლადი;
- ნებისმიერი ცვლადს, რომელიც გამოცხადებულია ფუნქციების გარეთ ეწოდება ***გლობალური ცვლადი*** - პროგრამის გლობალური ცვლადი;

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

int main()
    {
        //ლოკალური ცვლადები
        int i, j;
        int k;

        //ლოკალური ცვლადების ინიციალიზაცია
        i=10;
        j=20;
        k=i+j;

        cout<<k<<endl;

        return 0;
    }
```

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

//გლობალური ცვლადი
int k;

int main()
    {
        //ლოკალური ცვლადები
        int i, j;

        i=10;
        j=20;
        k=i+j;

        cout<<k<<endl;

        return 0;
    }
```

ლოკალური ცვლადების უპირატესობა

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

//გლობალური ცვლადი
int k = 45;

int main()
    {
        //ლოკალური ცვლადები
        int i, j, k=32;

        i=10;
        j=20;

        cout<<k<<endl;

        return 0;
    }
```

როდესაც ვსაზღვრავთ ლოკალურ ცვლადს ეს არ ნიშნავს რომ სისტემა მისთვის ავტომატურად გამოყოფს მეხსიერებას, ეს ჩვენ უნდა გავაკეთოთ შემდგომში ხელით - Heap Memory

გლობალური ცვლადის შემთხვევაში ცვლადისთვის მეხსერების სრულად გამოყოფა ხდება მაშინვე და როგორც წესი იწერება **0** ან **NULL** – Stack Memory

კონსტანტები

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

#define LENGTH 10
#define LENGTH 5
#define NEWLINE '\n'

int main()
    {
        int area;

        area = LENGTH * WIDTH;
        cout << area;
        cout << NEWLINE;

        return 0;
    }
```

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

int main()
    {
        const int LENGTH = 10;
        const int WIDTH = 5;
        const char NEWLINE = '\n';
        int area;

        area = LENGTH * WIDTH;
        cout << area;
        cout << NEWLINE;

        return 0;
    }
```