Школа рачунара и математике

www.rajak.rs
skolarajak@gmail.com programerska.radionica@gmail.com
www.facebook.com/Skola.Rajak.

Програмерска радионица

Програмски језик C

Додатни материјал

Аутори:

Ким Новак, Себастиан Новак

Предговор

Идеја овог писаног материјала је да прошири оно што је речено на предавањима, никада неће излазити ван оквира области за које је предвиђен, односно читање и усвајање и овог градива није неопходно за праћење наставе програмерске радионице.

Такође, на предавањима се неће тражити ништа што није на њима већ речено, како нико не би био оштећен.

Насупрот томе, при излагању градива у овим материјалима, подразумеваће се да је градиво са предавања усвојено и понављање градива са предавања ће бити минимално.

За оне који су заинтересовани и желе да науче више, овај материјал ће служити као помоћ при савладавању сложенијих проблема.

Све примедбе, критике, утиске, сугестије као и уочене грешке (лексичке или везане за градиво) можете послати мејлом на programerska.radionica@gmail.com .

Унапред захвални за Ваш труд:

Аутори

Садржај

6.0 Функције

Функције су начин паковања неког решења, у једну логичку целину, која се може посматрати независно и користити на различитим местима у реализацији главног решења.

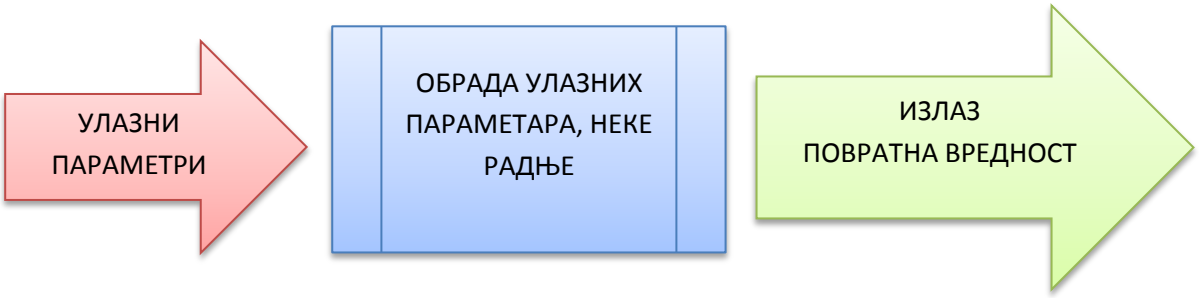
Паковањем кода у функцију, ми на један начин сакривамо како неки део нашег програма функционише. Условно речено сакривамо, јер ако неког занима може да узме да проучи цео код функције ако жели, него при гледању програма као једне велике слике.
На пример имамо неки део кода, одлуке неке и затим се на неком месту који представља неку сложенију обраду, само позива функција, која среди неке податке, на неки начин.
Пример:

```
int kub(int x); // funkcija koja vraca kub od prosledjene vrednosti (kub x-a je  x*x*x)

int main(){
  int x1=3,x2=4,x3=6;
 // pozivamo funkciju i dobijamo rezultat
// znamo kako je pozivamo i sta vraca
// ona izracunava kub onoga sto joj prosledimo
// ne znamo kako, ali ako ona radi kako treba, ni ne moramo da znamo
  x1 = kub(x2);
  x2= kub(x3);
}
```

Користили смо до сада готове функције из библиотеке стандардних функција stdio.h , морали смо знати како се функције позивају и шта оне уствари раде, бар идејно, да знамо коју када да користимо. Представљају за нас ЦРНЕ КУТИЈЕ.

Зашто црне кутије? Не видимо унутрашњост њену, већ само отворе у које можемо да убацимо неке параметре и можда да кроз неки други отвор добијемо неки резултат, ако функција враћа нешто. Ако кутија ради оно за шта нам треба, ми ћемо је користити. Ако желимо, увек имамо опцију да је пробамо отворити и погледати шта је унутра. Сад да ли ћемо имати довољно времена и стрпљења да у потпуности разумемо сваки њен део, је наш избор.

Тако, функције, у суштини, овако функционишу:



Као што смо на предавањима видели, функције тј. потпрограме, израђујемо у два корака:

1. Декларација функције – пишемо тип повратне вредности функције, дајемо јој име и пишемо листу њених параметара.
2. Дефиниција функција – пишемо тело функције, код који ће она извршавати.

Функије користимо тако што их позивамо. Да би схватили како се нека функција користи, морамо погледати њену декларацију. Баш због тога што декларације често служе само да покажу како се функција користи, зато се често називају и прототиповима функције.

6.1 Декларације функција

Декларација функције има више намена. Једна од њих је да представља прототип, скелет, инструкцију (упутство), за неког ко користи нашу функцију. Напише се каква се вредност враћа, да би неко могао да је преузме и пишу се параметри са којима се мора позвати.

Постоји разлог зашто пишемо декларације функција изнад главне функције.
Пошто је ово ипак почетнички курс, нећемо моћи ући у превише детаља иако је ово додатни материјал.
У програмском језику С имамо нешто што се зове досег важења променљиве и функција.
Досег важења одређује где и одакле је нека променљива или функција видљива.
Променљиве су видљиве од места блока у ком се декларишу, до краја тог блока. Блок је било који део кода унутар витичастих заграда { }.

Пример:

```
int main(){
    int x;
    // otvaramo novi blok koda, dozvoljeno je ovako otvarati, bez if-a, while ili neceg drugog
    // unutar novog bloka mogu biti nove deklaracije ali ce one postojati samo unutar bloka u kom su
    // definisane
    {
        int y;
        printf („x je %d , y je %d, x,y); // ovo ce raditi jer x vazi od pocetka do kraja main-a , ovo je deo
//main-a
    } // posle ove tacke, y vise ne postoji
    printf(„x = %d , y=%d“, x,y); // kompajler ce javiti gresku jer y se ne moze videti odavde, jer je unutar
//drugog bloka i ne postoji van njega

}
```

Слично важи и за функције. Функција се у коду може користити од места где је декларисана, па на доле. Тако да, ако би декларисали функције испод главне функције, мејна, не би их могли користити у мејну, јер њихов досег важења не би достизао до мејна.
Управо из овог разлога се #include наредбе пишу прве, на почетку кода, како би се све неопходне функције извадиле из .h фајлова и прекопирале у наш код, заједно са својим декларацијама и дефиницијама. То све компајлер одради за нас у току компајлирања. Пример неких функција и досега важења истих:

```
void f1(); // DEKLARACIJA - funkcija koja ispisuje hello World
int main(){
    f1(); // poziv funkcije f1, MEJN NE VIDI f2 !!!!!!!

}
void f2(); //DEKLARACIJA – FUNKCIJA JE VIDLJIVA ODAVDE PA NADALJE NA DOLE , iznad NE!
void f1(){
   // f1 vidi f2 i moze da je pozove
  printf("Hello World");
  f2();
}
void f2{
     printf("Mene vidi samo f1, jer sam deklarisana iznad f1, ne iznad main");
}
```

6.2 Дефиниције функција

Дефиниција функције представља писање тела функције. Давати код који ће та функције извршавати. Овде је врло важно нагласити пар ствари:
ПАРАМЕТРИ У ДЕФИНИЦИЈУ МОРАЈУ ДА БУДУ ИСТОГ ТИПА КАО И У ДЕКЛАРАЦИЈИ.
ПАРАМЕТРИ У ДЕФИНИЦИЈУ СЕ МОРАЈУ ИСТО ЗВАТИ КАО И У ДЕКЛАРАЦИЈИ.
ФУНКЦИЈА МОРА ИМАТИ ИСТИ ТИП ПОВРАТНЕ ВРЕДНОСТИ У ДЕФИНИЦИЈИ КАО ДЕКЛАРАЦИЈИ.

ПАРАМЕТРИ И СВЕ ОСТАЛЕ ПРОМЕНЉИВЕ УНУТАР ЈЕДНЕ ФУНКЦИЈЕ, НЕМАЈУ НИКАКВЕ ВЕЗЕ СА ПАРАМЕТРИМА ИЗ ГЛАВНЕ ФУНКЦИЈЕ ИЛИ БИЛО КОЈЕ ДРУГЕ ФУНКЦИЈЕ!!!
МОГУ ДА СЕ ИСТО ЗОВУ, АЛИ НЕ МОРАЈУ!!!
ПАРАМЕТРИ СУ САМО КОПИЈЕ ВРЕДНОСТИ КОЈЕ ДАМО ПРИ ПОЗИВУ!!!!!!!!

Дефиницију функцију пишемо испод мејна.
Дефиницију функције завршавамо са наредбом
return rez; // ili neki broj ili sta god vec ima vrednost istog tipa kao sto je u deklaraciji receno da ce biti
// tip povratne vrednosti
Ако функција не враћа ништа, односно она је типа void, довољно је написати само
return;

Примери дефиниција функција:

```
int kolicnik(int deljenik, int delioc);
void ispis();


int main(){
 int a=10,b=2,c;
c = kolicnik (a,b);
ispis();
}

int kolicnik(int deljenik, int delioc){
      // prvi nacin sa promenljivom za rezultat koji se vraca
      // int rez;
       // rez  =  deljenik / delioc;
      // return rez;
      // ili bez dodatne promenljive
      return deljenik / delioc;
}
void ispis(){
       int i;
       for(i=0; i<200;i ++){
            printf("%d");
       }
}
```

6.3 Паковање функција у библиотеке

Паковање функција у .h фајл се често ради, да би се постигао неки ред и прегледност главног програма. Главни програм само показује гране куда иде и како се све логичке целине уклапају, док се саме целине у неким посебним фајловима сакривене.

Стил лепог писања, односно прављења .h фајлова за смештање функција јесте да се пре било каквог кода направи докуметација, заглавље о опису библиотеке.

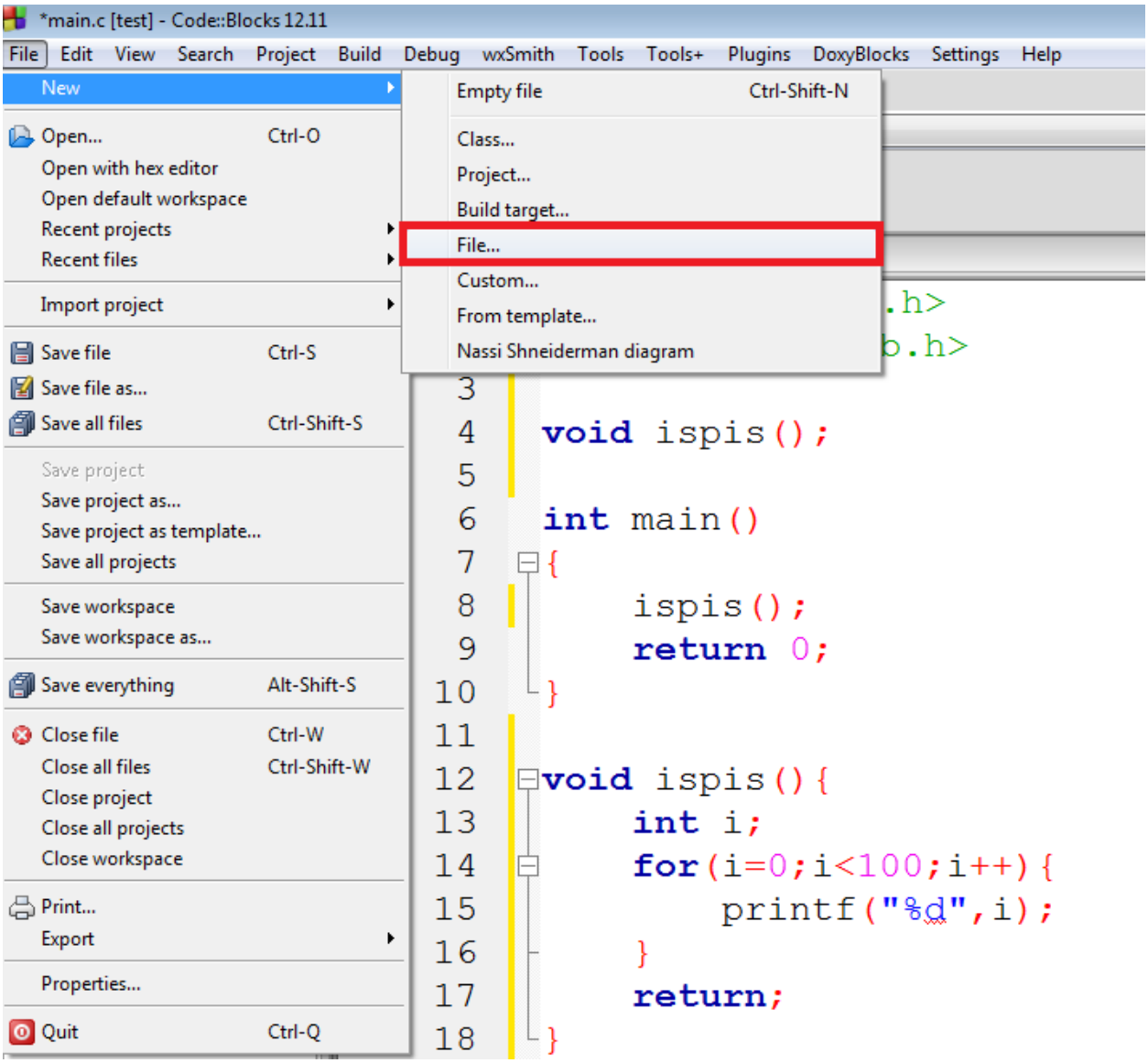
```
/*********************************
IME BIBLIOTEKE
SVRHA
Autor:
Datum:
Verzija:
Kontakt:
NAPOMENE:
Opis:

// opciono
Objasnjenje svake funkcije posebno
*********************************/
```

Затим следе речи које служе као заштита целог кода библиотеке. Заштита од вишеструког инклудовања, односно увезивања нпр. стандардних библиотека. Тиме се може и унутар мејна и унутар библиотеке писати #include <stdio.h>, без тога би компајлер одбијао да компајлира. Заштитна реч треба да буде јединствена, свака библиотека да има своје. Име библиотеке _ H_INCLUDED је један од начина да се осигура јединственост. Сав код библиотеке треба ставити унутар блока тих заштитних речи.

```
#ifndef BIBLIOTEKA_H_INCLUDED
#define BIBLIOTEKA_H_INCLUDED

 ////////////////////////////////////////////////////////
 //           DEKLARACIJE   FUNKCIJA                   //
////////////////////////////////////////////////////////

void mojaFunkcija(int k);
int zbir (int a, int b);

 ////////////////////////////////////////////////////////
 //           DEFINICIJE   FUNKCIJA                    //
////////////////////////////////////////////////////////

void mojaFunkcija(int k){
/*********************************
   Kratak opis funkcije, kako radi , itd
*********************************/
 //kod
}
int zbir (int a, int b){
 // OPIS i zatim kod
}

#endif // BIBLIOTEKA_H_INCLUDED
```

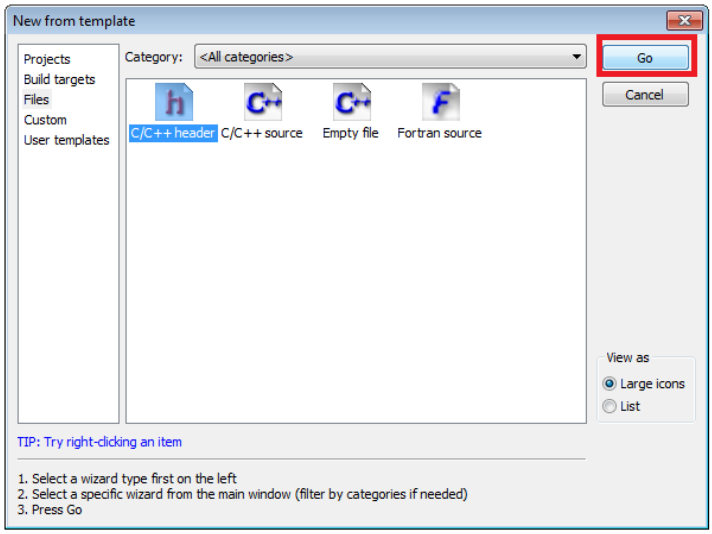
Ако .h ставите где Вам је и main.c, библиотеку тада можете у мејн увести, тако што напишете
#include "biblioteka.h"

Поступак за прављење библиотеке је приказан на следећим сликама:



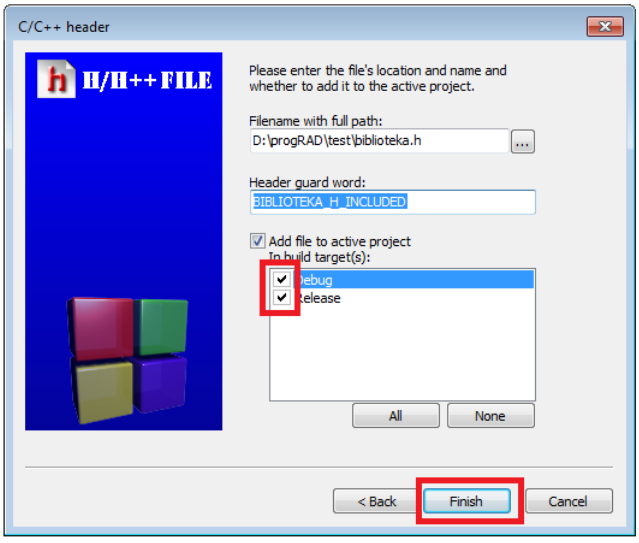
main.c

```c
#include <stdio.h>
#include <stdlib.h>

void ispis();

int main()
{
    ispis();
    return 0;
}

void ispis(){
    int i;
    for(i=0;i<100;i++){
        printf("%d",i);
    }
    return;
}
```



main.c

```c
#include <stdio.h>
#include <stdlib.h>

void ispis();

int main()
{
    ispis();
    return 0;
}

void ispis(){
    int i;
    for(i=0;i<100;i++){
        printf("%d",i);
    }
    return;
}
```



*main.c *biblioteka.h

```
7   Kontakt:
8   NAPOMENE:
9   Opis:
10
11  // opciono
12  Objasnjenje svake funkcije posebno
13  ***********************************/
14
15
16
17  #ifndef BIBLIOTEKA_H_INCLUDED
18  #define BIBLIOTEKA_H_INCLUDED
19
20  ////////////////////////////////////////////////////////////
21  //                                                        //
22  //                  DEKLARACIJE                           //
23  ////////////////////////////////////////////////////////////
24  void ispis();
25
26
27
28
29
30  ////////////////////////////////////////////////////////////
31  //                                                        //
32  //                   DEFINICIJE                           //
33  ////////////////////////////////////////////////////////////
34
35  void ispis(){
36      int i;
37      for(i=0;i<100;i++){
38          printf("%d",i);
39      }
40      return;
41  }
42
43  #endif // BIBLIOTEKA_H_INCLUDED
44
```

main.c biblioteka.h

```c
#include <stdio.h>
#include "biblioteka.h"

int main()
{
    ispis();
    return 0;
}
```