## PREDAVANJE 1

Main.java ------javac------> Main.class ------java-------->

*(prevajanje) (izvajanje)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

hello.c -------gcc-------> a.out ----------------->

*(prevajanje) (izvajanje)*

$ gcc hello.c

$ ./a.out

Hello world

JAVA -> Main.class vsebuje t.i. javansko bajtno kodo (java bytecode)

C -> ko se opravi prevajanje dobimo preveden program v strojnem jeziku

C zahteva da za vsako kombinacijo procesorja, operacijjskega sistema program vendo znova prevede

Torej a.out, ki deluje na tvojem računalniku ni nujno da deluje na drugem

$ gcc -o hello hello.c (izhod tega je datoteka hello, torej ni nujno da je vedno a.out – custom:))

$ ./hello

Hello world

V C-ju ni razredov in ni paketov. Zato nimamo nikolu public, static pri funkcijah… It’s shorter kot v javi

Ni C podoben javi, ampak je java podobna C-ju:)

Pri javi je ime publlic class-a enako imenu datoeke. Pri C-ju je ime vseeno

Ukaz echo $? vrne to, kar returna main funkcija

$ man printf

$ man exit

$ man getchar

ASCII – 7-biten

Table

Description automatically generated

# PREDAVANJE 3

## SPREMENJIVKE IN PODATKOVNI TIPI

CELOŠTEVILSKI TIPI

V javi je int predznačeno število (1 bit za predznak in 31 bitov za absolutno vrednost)

V C-ju: v intu imamo cela števila, ampak: koliko velika? Tega C ne pove, saj je int odvisen od tega na katerem procesorju program teče, ima pa določeno najmanjšo vrednost

char – vsaj 8 bitov -> signed char == char (default) – -27… + 27

unsigned char

int – vsaj 16 bitov -> short int

int

long int

(long long int)

signed int == int (default) – 1 bit za predznak, ostali predstavljajo vrednost

(od -231 do +231)

unsigned int – nima predznaka, ima samo absolutno vrednost (od 0 do 232 - 1)

Kdaj katerega izberem?

* V večini primerov izberem kar int
* Kadar hočem povedat, da so števila res samo 16-bitna ali hočem prihraniti prostor, uporabim short int – 2 mega pomnilnika
* Unsigned int bi lahko za indeksiranje tabele, saj so indeksi od 0 do n (ni negativnih)

Kako izvem kako velik je int?

* Pogledam dokumentacijo
* Napišem programček in potipamo kako velik je kakšen tip… npr poiščemo največje št unsigned inta in ko pride do preliva nehamo… Pri unsigned intu začnemo z ničlo in spet najdemo največje

Btw lahko napišem tudi unsigned int, unsigned short int itd.

Pazi pri izpisovanju podatkovnih tipov s funkcijo printf! %d je vredu za int, ampak za unsigned int uporabiš %u… itd

TIPI ZA ZAPIS S PLAVAJOČO VEJICO – FLOATING POINT

float

double

long double

* 1. s float ali z double ne moremo predstavit, lahko pa predstavimo število, ki je temu zelo blizu.
  2. = 1 \* 1/10 = 0 \* ½ + 0 \* ¼ + 0 \* 1/8 + 1 \* 1/16 + 1 \* 1/32 + …

(a + b) + c = a + (b + c) velja za realna števila. Vendar pa za float in double to ne velja vedno. Obstajajo protiprimeri.

Nauk zgodbe:

Floate ne primerjamo z “==”, temveč preverjamo, koliko sta števili blizu

**printf:**

%d

%ld

%n

%ln

%f

%lf

LOGIČNI TIPI

#include <stdbool.h>

bool b;

b = true;

b = false;

Če gre to skozi prevajalnik, le-ta vidi tip \_Bool b; in b = 1 ter b = 0

True in false veljata kot 0 in 1

LOGIČNI OPERATORJI

&&, ||, !

a && b

OPERATORJI NA BITIH

&, |

Primer:

int x = 3 -> 11

int y = 1 -> 1

x & y:

01

Sklad

Kopica

a.out

Statične spremenljivke (in konstante)

Strojna koda programa

# PREDAVANJE 4

KAZALCI

Kazalec je spremenljivka, ki vsebuje naslov (spremenljivke/funkcije).

Sklad

Kopica

NASLOVI

FFFF……………………………………………0

Statične spremenljivke (in konstante)

Strojna koda programa

KAZALČNI TIP:

**int \*pi;** //pi = kazalec na int

**char \*pc;** // pc = kazalec na char

**double \*pd;** // pd = kazalec na double

pi se naloži na stack, int i pa na mesto, kjer so spravljene statične spremenljivke.

DVA OPERATORJA:

**&** - vrne naslov

**\*** - usmeri na tisto, kar je na naslovu

Če pri delu s kazalci nisi pazljiv, velikokrat dobiš napako v prevajalniku: segmentation fault

SCANF:

**scanf(“%d”, &i)** …. To dela, saj na naslov spremenljivke i nastaviš prebrano vrednost in scanf piše na stack, kjer se nahaja tudi naslov i-ja.

**i = 0;**

**scanf(“%d”, i)**…. V funkcijo scanf pošljem v tem primeru samo vrednost i-ja, torej ničlo. Ne dela, saj potem scanf razume to kot naslov 0

TABELE 1D

V C-ju nimamo funkcije length, zato ne moremo vedeti koliko velika je tabela, razen če imamo velikost shranjeno v spremenljivki.

Tabele so na skladu, zato (če imamo velike) jih ne smemo dat na sklad.