Sveučilište u Zagrebu
PMF – Matematički odsjek

Objektno programiranje (C++)

Predavanja 10 - GMP

Vinko Petričević

GMP

- Za rad sa velikim brojevima u C/C++ postoje brojne biblioteke
- Na ovim predavanjima ćemo se malo dotaknuti librarya GMP https://gmplib.org/
- Pod Linuxom dolazi u standardnom paketu, a može se i skinuti kod, te samostalno kompajlirati
- Pod Visual Studiom već nekoliko verzija nije ugrađena podrška (prije samo otvorite projekt, te ga kompajlirate), ali za potrebe testiranja radi automatski pod npr. Cygwin-om
- To je prije svega C-library (dobar dio funkcija je pisan u assembleru, optimiziran za velik broj različitih procesora), a ima i jako lijepo napisane C++ klase
- #include <gmp.h> odnosno <gmpxx.h>
- gcc program.c -lgmp
  g++ program.cpp -lgmp -lgmpxx

GMP – Covski tipovi

- Za rad sa cijelim brojevima imamo strukturu mpz_t, za rad sa racionalnim mpq_t, dok za rad s decimalnim brojevima možemo koristiti mpf_t
- Funkcije za rad s pojedinim tipovima uglavnom počinju imenom tipa, npr. mpz_..., mpq_... Ili mpf_..., a imamo i funkcije koje rade s prirodnim brojevima mpn_...
- Npr. mpz_mul(c, a, b) će izračunati a*b i rezultat spremiti u c
- Prije korištenja svake strukture moramo napisati mpz_init(n), a na kraju mpz_clear(n);
- Interno, svaka struktura ima u sebi pokazivač na proizvoljno velik niz znamenaka (mp_limb_t)
- Primjer – factorial.c
- Parametri se funkcijama šalju tako da se prvo stavlja rezultat, pa onda operandi
- Funkcije koje sami pišemo također ne bi trebale vraćati mp?_ stukture, nego ih vraćati kroz parametar
- Za neke funkcije imamo više verzija, koje nam mogu ubrzati program u nekim situacijama
- Npr. Ukoliko je broj kojim množimo potencija broja 2, brže će raditi mpz_mul_2exp, od mpz_mul (kao i shiftanje kod običnih brojeva)
- Ukoliko znamo da broj koji koristimo kao parametar jedna znamenka (64 ili 32-bitni broj, zavisi o prevodiocu), bolje je koristiti funkcije s nastavkom _ui ili _si, npr. mpz_add_ui
- Neke funkcije su brže kada koristimo isti parametar kao povratnu vrijednost (+=), a neke nisu (*=)

GMP – Covski tipovi

- Ukoliko koristimo racionalne brojeve, pri inicijalizalizaciji sami trebamo osigurati da su brojnik i nazivnik skraćeni. Nakon toga će nakon svake operacije i ostati skraćeni
- Brojnik i nazivnik možemo dobiti sa makroima mpq_numref i mpq_denref

GMP – Covske najčešće funkcije

- mpz_init (mpz_t x) – inicijalizira varijablu, i postavi joj vrijednost na 0
- mpz_inits – inicijalizira null-terminated listu varijabli i postavi im vrijednost na 0
- mpz_init2(mpz_t x, broj) – inicijalizira varijablu i zauzima prostor da stane broj bitova i postavi vrijednost na 0
- mpz_realloc2(mpz_t x, broj)
- mpz_clear, mpz_clears
- mpz_set, (_ui, _si, _d, _q, _f, _str) postavlja vrijednost, s tim da u _str verziji još imamo i bazu
- Kod svih funkcija broj prvo mora biti init-an, a imamo i funkcije koje rade oboje, npr. mpz_init_set, …, mpz_init_set_str
- mpz_swap
- mpz_get (…), konvertiramo varijablu u C-ovski tip
- mpz_add, mpz_add_ui
- mpz_sub, mpz_sub_ui, mpz_ui_sub
- mpz_mul, mpz_mul_ui, mpz_mul_si
- mpz_addmul, mpz_addmul_ui, mpz_submul, mpz_submul_ui
- mpz_mul_2exp
- mpz_neg
- mpz_abs

GMP – Covske najčešće funkcije

- mpz_cdiv_q, _r, _qr i još svaka sa podvlakom _ui _2exp
- Osim cdiv-a, imamo i fdiv sa svim opcijama, te tdiv. S c rezultat zaokružuje na gore (ceil, ostatak će biti negativan), f ne dolje (floor), a t dođe od truncate, ali svakako će vrijediti n=qd+r, i r i d po apsulutnoj vrijenosti zadovoljavaju teorem o dijeljenju s ostatkom

- mpz_mod, mpz_divexact, mpz_divisible_p, mpz_congruent_p

- mpz_cmp (…)

- mpz_and, _ior, _xor, _com

- mpz_powm (rez, baza, exp, modulo) mpz_pow(rez, baza, exp)
- mpz_root, mpz_root_rem, mpz_sqrt (_rem)
- mpz_perfect_power_p, mpz_perfect_square_p

- mpz_gcd, mpz_lcm, mpz_gcdext, mpz_invert
- mpz_next_prime

GMP – Covske najčešće funkcije

- Možemo i učitavati/ispisivati brojeve u datoteku
- Postavljati/očitavati određenu znamenku
- mpz_size vraća broj znamenki

- Sve slične funkcije imamo i za rad s racionalnim (i sa decimalnim) brojevima
- mpq_numref, (_denref) dobijemo referencu na brojnik/nazivnik
- mpq_get_num (_den) mpz_set_num (_den) očitavamo/postavljamo brojnik/nazivnik

- Kod decimalnih brojeva imamo funkcije mpf_set_default_prec ili _get za postavljanje minimalne preciznosti brojeva. Nakon ovoga svaka sljedeća mpf_init funkcija će alocirati dovoljno potrebne memorije, dok mpf_set_prec postavlja preciznost određene varijable

- gmp_printf, fprintf, sprintf ispisuju standardno brojeve. Iza % pišemo Z, Q ili F
- sa gmp_scanf ih možemo učitavati (ne treba referencu pisati)

- Primjeri sa razlomci i decimal

GMP – C++

- Osim svega do sada nabrojanog, u biblioteci su napravljene i C++-ovske klase koje koriste prethodne funkcije, te su operatori napravljeni da rade veoma efikasno

- Imamo klase mpz_class, mpq_class i mpf_class
- Na svakoj klasi imamo i funkcije get_mpX_t(), kojima dobivamo C-ovsku klasu (X=z, q ili f), npr. mpz_gcd(a.get_mpz_t(), b.get_mpz_t(), c.get_mpz_t()); a = gcd(b, c);

- Operatori su napravljeni tako da se nepotrebna kopiranja memorije što više izbace, npr.
- a=b+c će rezultirati time da će se zbrajanje računati tek na operatoru pridruživanja

GMP – alokacija memorije

- Sve strukture/klase mogu raditi sa proizvoljno velikim brojevima (koliko god imamo memorije na računalu)
- Po defaultu će koristiti normalne funkcije alloc/realloc/free
- Ali te funkcije možemo jednostavno promijeniti sa
  void mp_set_memory_functions (
  void *(*alloc_func_ptr) (size t),
  void *(*realloc_func_ptr) (void *, size t, size t),
  void (*free_func_ptr) (void *, size t))
  ali to moramo napraviti prije nego što je aktivan ijedan GMP objekt
- Isto tako imamo i mp_get_memory_functions