**Tugas Kelompok ke-1**

**Week 2**

* **Muhammad Rizky Fadli – 2502121175**
* **Wulan Aprianingsih – 2502126554**
* **Frans Sebastian – 2502121162**
* **Siti Dahlia -** **2502126983**

Jawablah soal berikut dengan benar!

1. Soal Essay
2. Berikan penjelasan dan penerapannya dalam program untuk hal dibawah ini
* Argument

Argument adalah nilai yang diberikan kepada fungsi saat dipanggil. Nilai ini digunakan oleh fungsi untuk melakukan operasi atau menghitung hasil. Contoh penerapannya ketika kita memesan kopi di kedai. Kita bilang, “Saya ingin kopi arabika, dengan susu.” Di sini, “kopi arabika” dan “dengan susu” adalah argument yang kita berikan kepada pelayan (fungsi) untuk membuat pesanan.

* Parameter

Parameter adalah variabel yang digunakan untuk menyimpan nilai argument dalam fungsi. Parameter didefinisikan dalam deklarasi fungsi. Menggunakan contoh pada penjelasan argument di atas, pelayan (fungsi) akan menerima pesanan kita. Dia akan mengingat bahwa kita mau “kopi arabika” dan “dengan ssu”. Di sini, “kopi arabika” dan “dengan susu” adalah parameter yang digunakan pelayan untuk membuat pesanan kita

Contoh penggunaan argument & parameter :

|  |
| --- |
| * #include <iostream>
* using namespace std;
* // Fungsi 'orderkopi' dengan parameter 'jenisKopi' dan 'susu'
* void orderkopi(string jenisKopi, bool susu) {
* cout << "Jamet memesan kopi " << jenisKopi;
* if (susu) {
* cout << " dengan susu.";
* } else {
* cout << " tanpa susu.";
* }
* }
* int main() {
* // Memanggil fungsi 'orderkopi' dengan argument "arabika dan false
* orderkopi("arabika", true);
* return 0;

} |

|  |
| --- |
|  |

* Inline Function

Fungsi inline adalah jenis fungsi di mana kode fungsi tersebut disalin ke tempat pemanggilannya saat program dikompilasi, bukan melalui proses pemanggilan fungsi yang biasa. Hal ini berguna untuk meningkatkan kecepatan eksekusi terutama pada fungsi-fungsi kecil dan sering dipanggil. Namun, penggunaan inline function yang berlebihan dapat mengakibatkan pembesaran ukuran kode program. Contoh penerapan misal kita memiliki manual book untuk perbaikan atau perawatan kendaraan kita. Didalam manual book itu tercantum semua petunjuk pekerjaan yang harus dilakukan. Nah untuk menghemat waktu kita buat program yang akan memanggil catatan tersebut, misal saja kita ingin mengganti aki motor. Analoginya, fungsi inline bertindak seperti secarik kertas yang ditempelkan di tempat pemanggilannya, mempercepat proses eksekusi tanpa harus melakukan panggilan fungsi secara terpisah. Berikut contoh kodenya

|  |
| --- |
| #include <iostream>using namespace std;// Misalkan ini adalah manual book kendaraan motor mio smile yang berisi banyak petunjuk// Fungsi inline 'menghidupkanmesin' motor mio smileinline void menghidupkanmesin() { cout << "1. Masukan kunci kedalam socket kunci.\n"; cout << "2. Putar kunci kekanan.\n"; cout << "3. Tekan tuas rem sambil menekan tombol starter.\n"; cout << "4. Motor sudah hidup.\n";}// Fungsi inline 'gantiaki' motor mio smileinline void gantiakimio() { cout << "1. Siapan obeng kembang dan kunci ring ukuran 10.\n"; cout << "2. Buka jok belakang.\n"; cout << "3. Buka penutup aki yang berada di dekat bagasi.\n"; cout << "4. Setelah terbuka copot baut penghubung aki dan kabel.\n"; cout << "5. Setelah kabel tercopot angkat dan keluarkan aki tersebut.\n"; cout << "6. Lalu masukan aki baru kedalam ruang aki tersebut\n"; cout << "7. Setelah itu pasang kembali kabel dan baut lalu kencangkan dengan kunci ukuran 10\n"; cout << "8. Setelah kabel terpasang, lanjutkan dengan menutup tempat aki dengan tutupnya\n"; cout << "9. Setelah itu coba hidupkan motor kembali\n"; cout << "10. Selesai dan motor siap dipakai kembali \n";}int main() { // Memanggil fungsi inline 'gantiaki' motor mio smile gantiakimio(); return 0;} |

|  |
| --- |
|  |

* Friend Functions

Friend function adalah fungsi yang didefinisikan di luar suatu kelas tetapi memiliki hak untuk mengakses semua anggota (private dan protected) dari kelas tersebut. Contoh penerapan misalkan ada 4 orang, yaitu Rizky, Wulan , Siti & Frans. Rizky merupakan anggota klub sepeda, sedangkan Wulan, Siti dan Frans bukan merupakan anggota. Lalu Rizky bisa masuk kedalam ruangan klub tersebut meskipun bukan merupakan pemilik (bukan ketua dan sejenisnya) tetapi bisa membuka ruangan karena termasuk anggota dari klub tersebut.

* Friend Class

Friend class adalah kelas yang memiliki hak untuk mengakses private dan protected anggota dari kelas lain yang telah mendeklarasikannya sebagai teman.

Lalu disini Wulan dan Siti merupakan teman dari Rizky jadi mereka bisa masuk kedalam ruangan karena didalam klub tersebut mengizinkan membawa teman meskipun bukan anggota asalkan temannya dalam hal ini Rizky.

Contoh kode Friend funcitions dan friend class dengan simulasi contoh diatas :

|  |
| --- |
| #include <iostream>#include <string>using namespace std;class klubsepeda;class Anggota {public: string nama; Anggota(string n) : nama(n) {} void masukRuangan(klubsepeda& klub);};class klubsepeda {private: bool ruangan = true; // asumsikan ruangan awalnya terkuncipublic: friend class Anggota;};void Anggota::masukRuangan(klubsepeda& klub) { if (nama == "Rizky" || nama == "Wulan" || nama == "Siti") { klub.ruangan = false; // membuka ruangan cout << nama << " berhasil masuk ke ruangan klub sepeda.\n"; } else { cout << nama << " tidak memiliki akses ke ruangan klub sepeda.\n"; }}int main() { klubsepeda klub; Anggota rizky("Rizky"); Anggota frans("Frans"); Anggota wulan("Wulan"); Anggota siti("Siti"); // Memanggil fungsi 'masukRuangan' dari class 'Anggota' rizky.masukRuangan(klub); frans.masukRuangan(klub); wulan.masukRuangan(klub); siti.masukRuangan(klub); return 0;} |

|  |
| --- |
|  |

1. Berikan penjelasan anda kelebihan dan kekurangan bahasa C++ dari bahasa program lainnya serta penggunaan bahasa C++ saat ini.

(jika terdapat link website yang dijadikan acuan dapat dilampirkan)

**Jawab:**

Bahasa pemrograman C++ telah menjadi salah satu pilihan utama dalam dunia pengembangan perangkat lunak yang membutuhkan kinerja tinggi dan kontrol langsung terhadap perangkat keras. Keunggulan utamanya terletak pada kinerja tinggi yang dapat diandalkan dalam memproses data secara efisien, serta fleksibilitas yang memungkinkan penggunaan berbagai paradigma pemrograman seperti prosedural, berorientasi objek, dan generik. Dalam industri game, C++ telah menjadi tulang punggung pengembangan berkat kemampuannya dalam menangani logika permainan yang kompleks, grafika real-time yang memukau, dan interaksi langsung dengan perangkat keras grafis. Tak hanya itu, dalam pengembangan sistem operasi, C++ menjadi bahasa yang sangat relevan karena memungkinkan pengembang untuk mengakses dan mengendalikan perangkat keras secara langsung, serta memberikan kinerja yang stabil dan efisien.

Namun demikian, penggunaan C++ juga datang dengan sejumlah tantangan. Kompleksitas bahasa dan kebutuhan untuk mengelola memori secara manual dapat menjadi hambatan bagi pengembang, terutama bagi mereka yang baru memulai. Kesalahan memori, seperti kebocoran memori dan segfaults, seringkali merupakan masalah yang sulit diatasi dalam pengembangan perangkat lunak dengan C++. Meskipun ada upaya untuk mengatasi masalah ini dengan praktik terbaik dan alat bantu seperti sanitizers, risiko kesalahan tetaplah ada.

1. **Soal Case**
2. Carilah Penelitian menggunakan google scholar atau scopus atau media pencarian lagi yang valid untuk Penelitian. Penelitian yang menggunakan Bahasa c++ dengan range waktu 5 tahun kebelakang dari tahun sekarang. Minimal 3 artikel. Hal yang anda perlu cari adalah

**Jawab:**

1. **Artikel 1**
* **Judul:** Accelerating linear solvers for Stokes problems with C++ metaprogramming
* **Sumber**: <https://www.sciencedirect.com/science/article/abs/pii/S1877750320305809>
* **Problem:** Terbatasnya penerapan aliran incompressible pada metode elemen hingga Galerkin diskontinu karena biaya komputasi yang tinggi.
* **Solusi**: Teknik pemrograman C++ . Pemecah Pardiso langsung multithread dan pemecah PETSc iteratif paralel.
* **Metode/Design penelitia**n: Kualitatif dan Eksperimen.
* **Batasan penelitian**: Strategi prakondisi monolitik dan komposit untuk 3 masalah benchmark dan mengurangi jejak memori algoritme sekitar 40%.
* **Kesimpulan**: Efisiensi beberapa pemecah untuk sistem linier renggang besar yang diperoleh dari diskritisasi persamaan Stokes dengan metode elemen hingga Galerkin yang terputus-putus. Seperti yang diharapkan, pemecah Pardiso langsung dari MKL menunjukkan skalabilitas terburuk baik dari segi waktu solusi dan jejak memori solusi.
* **Saran penelitian selanjutnya**: Menampilkan contoh program atau code yang digunakan untuk menyelesaikan masalah aliran incompressible pada metode elemen hingga Galerkin diskontinu.
1. **Artikel 2**
* **Judul**: CRCount: Pembatalan Pointer dengan Penghitungan Referensi untuk Meredakan Kesalahan Penggunaan Setelah Bebas pada Kode Legacy C/C++
* **Sumber**: Jangseop Shin, Donghyun Kwon, Jiwon Seo, Yeongpil Choy, dan Yunheung Paek dari Universitas Nasional Seoul, serta Yeongpil Choy dari Universitas Soongsil.
* **Permasalahan**: Kesalahan penggunaan setelah bebas (Use-after-free/UAF) merupakan ancaman keamanan yang signifikan dalam perangkat lunak, terutama pada basis kode legacy C/C++, karena sulitnya mengidentifikasi dan meredakan mereka secara efektif. Pendekatan yang sudah ada seperti pembatalan pointer eksplisit mengalami overhead kinerja dan memori yang tinggi.
* **Solusi**: Makalah ini mengusulkan CRCount, sebuah solusi untuk meredakan kesalahan UAF dalam basis kode legacy C/C++. CRCount menggunakan pembatalan pointer implisit yang diperkuat oleh penghitungan referensi, dengan tujuan mencegah pointer yang menggantung dengan hanya membebaskan objek saat tidak ada referensi yang mengarah kepadanya.
* **Metode/Desain penelitian**: CRCount menggunakan teknik yang disebut penandaan pointer untuk menghitung referensi dengan akurasi tinggi bagi objek yang dirujuk oleh pointer dalam kode legacy C/C++. Teknik ini menggunakan struktur data yang disebut bitmap pointer, yang diperbarui melalui instrumen program dan perpustakaan runtime.
* **Batasan penelitian**: Penelitian ini berfokus pada meredakan kesalahan UAF khususnya dalam basis kode legacy C/C++. Solusi yang diusulkan, CRCount, diimplementasikan sebagai langkah kompilator di LLVM, memberikan aplikabilitas yang luas untuk program C/C++.
* **Kesimpulan**: CRCount secara efektif mengatasi masalah UAF dalam basis kode legacy C/C++ dengan memanfaatkan pembatalan pointer implisit melalui penghitungan referensi. Hasil empiris menunjukkan kemampuannya untuk melacak pointer dengan akurasi relatif rendah overhead dan meredakan kesalahan UAF dengan overhead kinerja dan memori yang lebih rendah dibandingkan solusi yang sudah ada.
* **Saran penelitian** selanjutnya: Penelitian mendatang dapat mengeksplorasi optimisasi dan penyempurnaan lebih lanjut pada CRCount untuk meningkatkan kinerja dan akurasi dalam menangani kesalahan UAF dalam berbagai basis kode C/C++. Selain itu, penyelidikan integrasi CRCount ke dalam kerangka kompilator atau alat pengembangan lainnya dapat meningkatkan aksesibilitas dan kegunaannya bagi para pengembang.
1. **Artikel 3**
* **Judul**: Verifikasi Program C++ menggunakan Model Checking
* Sumber: Penelitian oleh Felipe R. Monteiro, Mikhail R. Gadelha, dan Lucas C. Cordeiro dari Federal University of Amazonas, Brazil; Igalia, A Coruña, Spain; dan University of Manchester, Manchester, UK.
* **Problem**: Masalah keamanan dan kesalahan memori dalam bahasa pemrograman sistem seperti C atau C++ telah menjadi salah satu sumber kerentanan keamanan yang signifikan. Namun, verifikasi program C++ masih terbatas dan sulit dilakukan dengan sukses.
* **Solusi**: Penelitian ini mengusulkan pendekatan verifikasi baru berbasis model checking terbatas (BMC) dan satisfiability modulo theories (SMT) untuk memverifikasi program C++.
* **Metode/Design penelitian**: Penelitian ini menganalisis program C++ terbatas dengan mengkodekannya ke dalam SMT berbagai fitur canggih yang ditawarkan oleh bahasa pemrograman C++, seperti templates, inheritance, polymorphism, exception handling, dan Standard Template Libraries (STL). Fitur-fitur ini diformalkan dalam kerangka verifikasi formal menggunakan fragmen logika orde pertama yang dapat diputuskan.
* **Batasan penelitian**: Penelitian ini fokus pada verifikasi program C++ terbatas menggunakan pendekatan BMC dan SMT.
* **Kesimpulan**: Pendekatan verifikasi yang diusulkan dapat menangani sejumlah besar program C++, memberikan hasil verifikasi yang lebih tinggi dibandingkan dengan pendekatan lain, dan berhasil mendeteksi kesalahan aritmatika overflow dalam aplikasi komersial C++ di domain telekomunikasi.
* **Saran penelitian selanjutnya**: Penelitian selanjutnya dapat memperluas cakupan verifikasi untuk fitur-fitur tambahan dalam bahasa C++, serta mengoptimalkan kinerja dan efektivitas pendekatan verifikasi yang diusulkan.