PERTEMUAN 1

DESAIN PROGRAM

OBJEKTIF

STEPS IN PROGRAM DEVELOPMENT

- Mendeskripsikan langkah-langkah dalam proses pengembangan program

PROGRAM DESIGN METHODOLOGY

- Memperkenalkan metodologi desain program

PROCEDURAL VERSUS OBJECT-ORIENTED PROGRAMMING

- Memperkenalkan pemrograman prosedural dan berorientasi objek

AN INTRODUCTION TO ALGORITHMS AND PSEUDOCODE

- Memperkenalkan algoritma dan pseudocode

PROGRAM DATA

- Mendeskripsikan data

Tiga pendekatan yang berbeda untuk desain program diperkenalkan, yaitu prosedur-driven, event-driven dan desain program data-driven. Pemrograman prosedural dan pemrograman berorientasi objek diperkenalkan, bersama dengan pengembangan top-down dan desain modular.

Algoritma didefinisikan sebagai seperangkat instruksi yang terperinci, tidak ambigu, dan berurutan yang dikembangkan untuk menggambarkan proses yang diperlukan untuk menghasilkan output yang diinginkan dari masukan yang diberikan.

Pseudocode adalah cara seperti bahasa Inggris untuk mewakili algoritma. Pemrogram/programmer perlu memiliki pemahaman yang baik tentang data yang akan diproses;oleh karena itu, variabel data, konstanta dan literal didefinisikan, dan dasar-dasar data, struktur data, file dan validasi data.

STEPS IN PROGRAM DEVELOPMENT

"Computer programming is an Art".

Pemrograman dapat didefinisikan sebagai pengembangan solusi untuk masalah yang telah diidentifikasi sebelumnya, merangkai instruksi terkait, dan ketika 'dipindahkan' ke komputer, akan menghasilkan hasil yang diinginkan.

Ini bagian pertama dari definisi programming adalah untuk merancang solusi untuk masalah yang diidentifikasi.

Melompat langsung ke fase pengkodean tanpa mendesain terlebih dahulu solusi yang tepat biasanya menghasilkan program yang mengandung banyak kesalahan sehingga seringkali programmer kemudian menghabiskan banyak waktu untuk menemukan dan memperbaiki kesalahannya.

Seorang programmer yang berpengalaman akan terlebih dahulu merancang solusi terlebih dahulu, cek solusi secara manual (desk check), baru pindahkan/tuliskan ke kode program bahasa pemrograman yang dipilih.

STEPS IN PROGRAM DEVELOPMENT

Tujuh langkah dasar pengembangan program:

1. Define the problem.
2. Outline the solutions.
3. Develop the outline into an algorithm.
4. Test the algorithm for correctness.
5. Code the algorithm into a specific programming language.
6. Run the program on the computer.
7. Document and maintain the program

STEPS IN PROGRAM DEVELOPMENT

1. Define the problem (Menentukan Masalah)

Langkah ini melibatkan **membaca dan membaca ulang masalah dengan cermat** sampai Anda benar-benar memahami apa yang diperlukan.

Untuk membantu analisis awal ini, masalah harus dibagi menjadi tiga komponen terpisah:

- Masukan (the Inputs)
- Keluaran (The Outputs)
- Langkah-langkah pemrosesan untuk menghasilkan output yang dibutuhkan

STEPS IN PROGRAM DEVELOPMENT

2. Outline the solution. Outline solusi setelah masalah didefinisikan, Anda dapat memutuskan untuk memecahnya menjadi tugas atau langkah yang lebih kecil, dan membuat garis besar (outline) solusi.

Garis besar awal ini biasanya merupakan rancangan kasar dari solusi dan dapat mencakup:

- langkah-langkah utama yang terlibat
- subtugas utama (jika ada)
- antarmuka pengguna (jika ada)
- struktur kontrol utama (misalnya loop pengulangan)
- variabel utama dan struktur record
- logika utama.

STEPS IN PROGRAM DEVELOPMENT

3. Develop the outline into an algorithm
(Kembangkan garis besar menjadi algoritma).

Garis besar solusi yang dikembangkan pada Langkah 2 kemudian dikembangkan menjadi algoritma:

Serangkaian langkah-langkah tepat yang menggambarkan tugas-tugas yang harus dilakukan dan urutan pelaksanaannya. Menggunakan pseudocode (bentuk bahasa Inggris terstruktur) untuk mewakili algoritma solusi.

STEPS IN PROGRAM DEVELOPMENT

4. Test the algorithm for correctness (Uji kebenaran algoritma)

Langkah ini adalah salah satu yang paling penting dalam pengembangan program, namun ini adalah langkah yang paling sering dilewati.

Tujuan utama dari desk checking algoritma adalah untuk mengidentifikasi kesalahan logika utama lebih awal, sehingga kesalahan dapat dengan mudah dikoreksi.

Data uji perlu berjalan melalui setiap langkah dalam algoritma untuk memeriksa bahwa instruksi dijelaskan algoritma akan benar-benar melakukan apa yang seharusnya. Pemrogram ‘berpikir' melalui logika algoritma, persis seperti komputer, melacak semua variabel utama pada selembar kertas.

STEPS IN PROGRAM DEVELOPMENT

5. Code the algorithm into a specific programming
(Kodekan algoritma ke dalam pemrograman)

Kodekan algoritma ke dalam pemrograman pilihan. Bahasa pemrograman hanya setelah semua pertimbangan desain dalam empat langkah sebelumnya terpenuhi, Anda harus benar-benar mulai mengkodekan program ke dalam pemrograman pilihan Anda.

STEPS IN PROGRAM DEVELOPMENT

6. Run the program on the computer (Jalankan program di komputer)

Langkah ini menggunakan kompiler program dan data uji yang dirancang pemrogram untuk menguji kode kesalahan sintaksis (yang terdeteksi pada waktu kompilasi) dan kesalahan logika (yang terdeteksi pada waktu berjalan).

Jika program telah dirancang dengan baik, frustrasi dan keputusasaan yang sering dikaitkan dengan pengujian program dapat dikurangi seminimal mungkin. Langkah ini mungkin perlu dilakukan beberapa kali hingga Anda yakin bahwa program berjalan sesuai kebutuhan.

STEPS IN PROGRAM DEVELOPMENT

7. Document and maintain the program (Dokumentasikan dan pemeliharaan program)

Dokumentasi program tidak boleh dicantumkan sebagai langkah terakhir dalam proses pengembangan program, karena ini benar-benar merupakan tugas berkelanjutan dari definisi awal masalah hingga hasil tes akhir.

Dokumentasi mencakup:

a. Dokumentasi eksternal (seperti bagan hierarki, algoritma solusi, dan hasil data pengujian) dan

b. Dokumentasi internal yang mungkin telah dikodekan dalam program.

Pemeliharaan program mengacu pada perubahan yang mungkin perlu dilakukan pada program. Seringkali, perubahan ini dilakukan oleh programmer yang berbeda dari programmer yang pertama kali menulis program. Jika program telah dirancang dengan baik menggunakan teknik pemrograman terstruktur, kode akan terlihat sebagai dokumentasi diri, sehingga perawatan lebih mudah.

PROGRAM DESIGN METHODOLOGY

Prinsip dasar desain program didasarkan pada kenyataan bahwa sebuah program menerima data masukan, memproses data tersebut, dan kemudian mengirimkan data tersebut ke pengguna program sebagai keluaran.

Baru-baru ini, sejumlah pendekatan berbeda untuk desain program telah muncul, dan yang paling umum adalah:

- procedure-driven
- event-driven
- data-driven

PROGRAM DESIGN METHODOLOGY

Procedure-driven program design (Desain program yang digerakkan oleh prosedur)

Pendekatan prosedur-driven untuk desain program didasarkan pada gagasan bahwa fitur paling penting dari sebuah program adalah apa yang dilakukan oleh – proses atau fungsinya.

Berkonsentrasi pada apa yang harus dilakukan program, programmer mengidentifikasi dan mengatur proses dalam program. Aliran data masuk dan keluar dari setiap proses atau fungsi menjadi pertimbang. Strategi dikembangkan untuk memecah setiap fungsi menjadi aliran data yang lebih kecil dan lebih spesifik.

PROGRAM DESIGN METHODOLOGY

Event-driven program design (Desain program yang digerakkan oleh peristiwa)

Pendekatan event-driven untuk desain program didasarkan pada gagasan bahwa suatu peristiwa atau interaksi dengan dunia luar dapat menyebabkan program berubah dari satu keadaan ke keadaan yang lain.

Status identifikasi awal suatu program, kemudian semua pemicu yang mewakili peristiwa yang valid untuk status tersebut ditetapkan. Misalnya, ketika pengguna program dapat saja memilih “mouse event”mengklik tombol kiri mouse, mengklik tombol kanan mouse, menyeret mouse atau mengklik dua kali mouse, maka setiap tindakan klik mouse tersebut dapat memicu perintah yang berbeda dalam program dan dengan demikian tentukan akan menghasilkan status program yang berbeda sesuai dengan “mouse event”.

PROGRAM DESIGN METHODOLOGY

Data-driven program design (Desain program berbasis data)

Pendekatan berbasis data untuk desain program didasarkan pada gagasan bahwa, data dalam suatu program lebih stabil daripada proses yang terlibat.

Pendekatan ini dimulai dengan analisis data dan hubungan antara data, untuk menentukan struktur data. Setelah struktur data didefinisikan, output/luaran data yang diperlukan diperiksa yang selanjutnya digunakan untuk menetapkan proses apa yang diperlukan untuk mengubah data input menjadi output/luaran yang diperlukan.

PROCEDURAL VS OBJECT-ORIENTED PROGRAMMING

Procedural Programming (Pemrograman prosedural)

Pemrograman prosedural didasarkan pada pendekatan terstruktur dan top-down untuk menulis program yang efektif. Pendekatan ini berkonsentrasi pada apa yang harus dilakukan oleh suatu program dan melibatkan pengidentifikasian dan pengorganisasian proses-proses dalam program. Masalahnya biasanya dipecah menjadi tugas atau fungsi yang terpisah dan mencakup:

- Pengembangan top-down (*the top-down development*),
- Desain modular (*modular design*).

PROCEDURAL VS OBJECT-ORIENTED PROGRAMMING

Top-down development

Dalam pengembangan desain program top-down, solusi umum untuk masalah diuraikan terlebih dahulu. Outline/Garis besar kemudian dibagi secara bertahap menjadi langkah-langkah yang lebih rinci sampai akhirnya tingkat yang paling rinci telah selesai.

Setelah proses pengembangan top-down (disebut juga dekomposisi fungsional atau stepwise refinement) programmer mulai membuat kode program.

Kelebihan dari desain pengembangan program menggunakan cara ini adalah ketepatan pemrograman yang lebih presisi daripada yang mungkin dilakukan sebelumnya.

PROCEDURAL VS OBJECT-ORIENTED PROGRAMMING

Modular design (desain modular)

Pemrograman prosedural juga menggabungkan konsep desain modular, yang melibatkan pengelompokan tugas bersama karena semuanya melakukan fungsi yang sama.

Desain modular terhubung langsung ke pengembangan top-down, karena langkah-langkah atau subtugas di mana solusi program dibagi membentuk modul program. Desain modular yang baik juga membantu dalam membaca dan memahami program.

PROCEDURAL VS OBJECT-ORIENTED PROGRAMMING

Object-oriented programming

Pemrograman berorientasi objek juga didasarkan pada pemecahan masalah; namun, fokus utamanya adalah pada hal-hal (atau objek) yang membentuk program.

Program ini berkaitan dengan bagaimana objek berperilaku, sehingga memecahkan masalah menjadi satu set objek terpisah yang melakukan tindakan dan berhubungan satu sama lain.

Objek-objek ini memiliki properti yang tetap, dan setiap objek bertanggung jawab untuk melakukan serangkaian tugas terkait.

AN INTRODUCTION TO ALGORITHM AND PSEUDOCODE

Sebuah program harus dirancang secara sistematis dan tepat sebelum pengkodean (menulis kode program) dimulai.

Proses desain ini menghasilkan konstruksi algoritma

AN INTRODUCTION TO ALGORITHM AND PSEUDOCODE

Apa itu algoritma?

Algoritma seperti resep: ia mencantumkan langkah-langkah yang terlibat dalam menyelesaikan tugas.

Algoritma dapat didefinisikan dalam istilah pemrograman sebagai seperangkat instruksi yang terperinci, tidak ambigu, dan teratur yang dikembangkan untuk menggambarkan proses yang diperlukan untuk menghasilkan output yang diinginkan, dari input yang diberikan.

Algoritma dapat ditulis dalam bahasa Inggris sederhana dan bukan dokumen formal. Namun, agar bermanfaat, ada beberapa prinsip yang harus dipatuhi. Sebuah algoritma harus memenuhi kriteria berikut:

- Jelas, tepat, dan tidak ambigu (bermakna ganda)
- memberikan solusi yang benar dalam semua kasus
- memiliki akhir.

Contoh algoritma

Turn on calculator
Clear calculator
Repeat the following instructions
    Key in dollar amount
    Key in decimal point (.)
    Key in cents amount
    Press addition (+) key
Until all prices have been entered
Write down total price
Turn off calculator

What is pseudocode?

Pseudocode merupakan cara populer untuk merepresentasikan algoritma. Ppseudocode telah dipilih sebagai metode utama untuk merepresentasikan suatu algoritma karena mudah dibaca dan ditulis, dan memungkinkan pemrogram untuk berkonsentrasi pada logika masalah.

Pseudocode adalah bahasa Inggris yang sangat terstruktur. Ini adalah bahasa Inggris yang telah diformalkan dan disingkat agar terlihat seperti bahasa komputer tingkat tinggi. Tidak ada pseudocode standar saat ini. Penulis tampaknya mengadopsiteknik khusus mereka sendiri dan seperangkat aturan, yang sering kali menyerupai bahasa pemrograman tertentu.

Konvensi pseudocode:

1 Pernyataan ditulis dalam bahasa Inggris sederhana.

2 Setiap instruksi ditulis pada baris terpisah.

3 Kata kunci dan lekukan digunakan untuk menandakan struktur kontrol tertentu.

4 Setiap set instruksi ditulis dari atas ke bawah, dengan hanya satu entri dan satu luaran.

5 Kelompok pernyataan dapat dibentuk menjadi modul, dan modul itu diberi nama.

