***מדריך עיצוב קובץ***

|  |
| --- |
| 1. **מטרה:**    1. יש להקפיד על מבנה אחיד כדי ליצור סדר וקריאות לכל קבצי הסימולטור. 2. **מבנה:**    1. כל קובץ יתחיל עם הערה שמתארת מה מטרת הקובץ.    2. כל קובץ **.h** יתחיל עם **#pragma once** כדי למנוע **include** יותר מפעם אחת.    3. יש לאסוף מכל רחבי הקובץ את ה-**#include** רק של המערכת, למיין לפי סדר שימוש ולמחוק כפילויות.    4. לפני רשימה זו יש לשים את הכותרת **ספריות מערכת** **//**.    5. לאחר רשימה זו יש לשים את הכותרת **סוף ספריות מערכת** **//**.    6. יש לאסוף מכל רחבי הקובץ את כל שאר ה-**#include**, למיין לפי סדר שימוש ולמחוק כפילויות.    7. לפני רשימה זו יש לשים את הכותרת **ספריות סימולטור** **//**.    8. לאחר רשימה זו יש לשים את הכותרת **סוף ספריות סימולטור** **//**.    9. יש לאסוף מכל רחבי הקובץ את הגדרות קדם המעבד (**Preprocessor**) ולמחוק כפילויות.    10. יש למזג בלוקים של **#ifdef** **DEBUG** לדוגמא.    11. לפני רשימה זו יש לשים את הכותרת **הנחיות קדם מעבד** **//**.    12. לאחר רשימה זו יש לשים את הכותרת **סוף הנחיות קדם מעבד** **//**.    13. יש לאסוף מכל רחבי הקובץ את הגדרות הקבועים הגלובליים, למיין לפי סדר שימוש ולמחוק כפילויות.    14. לפני רשימה זו יש לשים את הכותרת **קבועים גלובליים** **//**.    15. לאחר רשימה זו יש לשים את הכותרת **סוף קבועים גלובליים** **//**.    16. יש לאסוף מכל רחבי הקובץ את הכרזות המשתנים הגלובליים, למיין לפי סדר שימוש ולמחוק כפילויות.    17. לפני רשימה זו יש לשים את הכותרת **הכרזות** **משתנים גלובליים** **//**.    18. לאחר רשימה זו יש לשים את הכותרת **סוף הכרזות משתנים גלובליים** **//**.    19. יש לאסוף מכל רחבי הקובץ את הגדרות המשתנים הגלובליים, למיין לפי סדר שימוש ולמחוק כפילויות.    20. לפני רשימה זו יש לשים את הכותרת **הגדרות** **משתנים גלובליים** **//**.    21. לאחר רשימה זו יש לשים את הכותרת **סוף הגדרות משתנים גלובליים** **//**.    22. יש לאסוף מכל רחבי הקובץ את הכרזות הפונקציות הגלובליות, למיין לפי סדר שימוש ולמחוק כפילויות.    23. לפני רשימה זו יש לשים את הכותרת **הכרזות פונקציות //**.    24. לאחר רשימה זו יש לשים את הכותרת **סוף הכרזות פונקציות //**.    25. יש לאסוף מכל רחבי הקובץ את הגדרת הפונקציות ולמיין לפי סדר שימוש. פונקציות המנהלות שגיאות יהיו בסוף הבלוק הרלוונטי.    26. לפני רשימה זו יש לשים את הכותרת **הגדרת פונקציות** **//**.    27. לאחר רשימת פירוש הפונקציות יש לשים את הכותרת **סוף הגדרת פונקציות //**. זו גם צריכה להיות השורה האחרונה בקובץ. 3. **ריווח:**    1. יש להוסיף שורת רווח בודדת בין נושאים שונים בתוך אותה פונקציה או **scope**.    2. יש להוסיף 2 שורות רווח בין פונקציות, ובין מקטעי הקוד המפורטים בפסקה 2.    3. יש למחוק רווחים לא נכונים או מיותרים, כולל שורות.    4. יש לשים רווח בודד בין מלל ההערה לבין סימן ההערה כך: **// Remark**.    5. יש לשים רווח בודד בין טיפוס לבין מזהה כך: **int** **x;**.    6. קוד שמשמש אך ורק לאיתור שגיאות, ייכתב ללא רווחים מתחילת השורה, כדי שיבלוט יחסית לשאר הטקסט. 4. **הערות:**    1. יש לכתוב את ההערות בעברית במידת האפשר.    2. אם נדרשת הערה באנגלית, יש להקפיד על אות גדולה לפי כללי התחביר באנגלית.    3. יש להקפיד על איות וסגנון, כולל נקודה בסוף משפט.    4. אין להוסיף את שם המפתח בהערה, זה ניתן לבירור לפי מי שפרסם את ה-**commit**.    5. עדיף שהקוד יהיה ברור מאליו, כך שלא ידרשו הערות להסבירו.    6. יש לשים הערה לאחר כל פקודה סוגרת לדוגמא: **#endif // Define x**.    7. אין להשתמש בהערות של שפת **C**, אלא רק בהערות של **++C** כלומר **//**.    8. אין למחוק מקטעי קוד, אלא לבטל אותם ע"י הערות. רק אני אמחק מקטעים שלמים, אחרי שאהיה משוכנע שאין בהם צורך.    9. יש להוסיף הערה המציינת למה משמש כל קובץ **include** של המערכת. הרעיון הוא שאם כבר לא משתמשים בפונקציה מאותה ספריה, ניתן להפסיק להשתמש בכל הספרייה ולחסוך משאבים.    10. יש להוסיף הערה לפני כל מקטע קוד/מחלקה/פונקציה לגבי תפקידה.    11. יש להוסיף קישור לעזרה לכל פונקציה של מערכת ההפעלה או תכונה של השפה שבשימוש מועט. 5. **קיצורים:**    1. אם מזהה מכיל ראשי תיבות או קיצורים, אז הן יהיו באותיות גדולות לדוגמא **MFD\_button**.    2. מזהים של **define#**, קבועים ו-**enum** יהיו רק באותיות גדולות, עם קו תחתי בין מילים, גם אם מדובר בקיצורים או ר"ת.    3. מזהה לא יכיל שם של טיפוס או קיצורו, אלא אם זה משתנה זמני כגון **temp\_int** או טיפוס שהוא אינו חלק מהשפה.    4. שם של ספריית **include** יהיה עם אות גדולה היכן שזה קיצור לדוגמא **cISO646** ולא **ciso646**, כי אז קשה להבין שמדובר בר"ת של המילים **International Standard Organization**.    5. מותר **UserIDs**, כי **ID** זה ר"ת ידועים של **identification**. כנ"ל **TrackIR**. 6. **עיצוב:**    1. בהגדרת פונקציה וגם במימוש פונקציה עם מספר פרמטרים, יש לשים פרמטרים אחד מתחת לשני, מיושרים לפי הסוגריים הפותחים.    2. אין להגדיר מספר מזהים בשורה אחת. יש להפריד לשורות נפרדות ולאתחל כל משתנה בנפרד.    3. כל שורה תבצע פעולה לוגית אחת בלבד, גם במחיר של שימוש במשתנים זמניים עם **auto,** כדי לבצע פעולות מורכבות.    4. יש לכתוב פונקציות קטנות ומוגדרות מבחינה לוגית לפעולה אחת בלבד.    5. יש לפצל פונקציות ארוכות לפונקציות קטנות גם במחיר של העברת פרמטרים.    6. יש להצמיד את הכוכבית שמציינת מצביע (**pointer**) לטיפוס ולא לשם המשתנה.    7. יש להפוך את הסדר מ-**if (a == 0) ל-if (0 == a)** כדי לגלות שימוש באופרטור **=** במקום באופרטור **==**.    8. יש להשתמש ב- **\** ולא ב- **/** כדי לציין נתיב לתיקייה או קובץ.    9. יש להשתמש בסוגריים כדי למנוע אי הבנות בסדר פעולות חשבון.    10. אין להשתמש באופרטור **?:** יש להשתמש בשאלת **if**.    11. יש להשתמש בשמות אופרטורים לפי **iso646**. לדוגמא **or** במקום **||** ו-**not\_eq** במקום **=!**.    12. אם משהו ניתן למיון, עדיף למיין אותו מקטן לגדול כגון רשימת משתנים מאותו סוג. 7. **תחביר**:    1. אין לשנות את הגדרות התחביר כפי שהן מופיעות ב-**MSVS**. הגדרות אלו נבחרו בקפידה. בקשה לשינוי רק דרכי.    2. אין להשתמש ב-**#define**. עדיף **const**. כמו כן **enum** עדיף על סדרה של **const** ו-**enum class** עדיף על **enum**.    3. יש להשתמש ב-**constexpr** בכל ביטוי שתוצאתו ניתנת לחישוב בזמן הידור. זה עדיף על **const**.    4. יש להשתמש בקבועים, כולל בפרמטרים בפונקציות. אין להשתמש במספרים חסרי משמעות.    5. אין להשתמש בהמרות טיפוסים (**Type casting**) בסגנון **C**. מותר רק באמצעות מילים שמורות של **++C**.    6. יש להימנע משימוש ב-**typedef**.    7. אין להשתמש ב-**inline**. המהדר מוגדר לבצע זאת אוטומטית.    8. יש לוודא **delete** מקביל לכל **new** באותו **scope**.    9. יש לנסות להחליף **new** ו-**delete** בווקטורים סטטיים.    10. בהגדרת פונקציה שאינה מקבלת ערכים, יש להוסיף **void** בתוך הסוגריים כך: **(void)**.    11. יש להעביר לפונקציה טיפוסים פשוטים באמצעות ערך וטיפוסים מורכבים או גדולים באמצעות יחוס (**reference**).    12. יש להחליף מערכים פשוטים ב-**std::array** או עדיף בווקטור. 8. **מזהים ומשתנים**:    1. כל מזהה צריך להיות בפורמט **xxx\_yyy,** באותיות קטנות בלבד, לדוגמא **error\_code**.    2. לכל מזהה שלא שייך למערכת, לשפה או לספריות חיצוניות, כלומר שאנחנו הגדרנו, תהיה קידומת **my\_** לדוגמא: **my\_error\_code**.    3. מזהה שהוא מטיפוס שמוגדר ע"י המשתמש יהיה עם אות ראשונה גדולה, כלומר My\_.    4. קבועים ו-**define#** יהיו רק באותיות גדולות.    5. אין להשתמש ב-**using namespace std**. יש להשתמש ב-**using std::xxxxx**. זה נועד למנוע בלבול בשמות.    6. יש להגדיר משתנים מקומיים רק לפני שימוש ראשון. זה נועד לאיתור באגים.    7. אתחול משתנים יבוצע כך: **x = {7}**. זה נועד למנוע המרה לא רצויה ע"י המהדר. אופרטור ה-**=** הוא חובה כדי לאפשר לתבנית **sub\_range** לפעול ולגלות חריגים. משתנה מטיפוס **auto** יש לאתחל רק עם **=** וללא **{ }**.    8. חובה לאתחל כל אובייקט הניתן לאתחול. רצוי על ידי ערך לא חוקי או **{ }** אם אין אפשרות אחרת.    9. יש לאתחל משתנה לערך לא חוקי לפני שימוש חוזר.    10. משיקולי ביצועים, יש להשתמש ב-**bool**, **int** ו-**double**. לא ב-**short** או **float**.    11. יש לנסות להמיר **int** או **BOOL** ל-**bool,** אם משתמשים רק בשני מצבים.    12. יש להשתמש ב-**signed** ו-**unsigned** כדי למנוע אי הבנות. עדיף לא להשתמש ב-**unsigned**.    13. יש להעדיף **std::string** על **const char\*** היכן שאפשרי. מכיוון שמשתמשים ביוניקוד, עדיף **wstring** על **string**.    14. כל מצביע ישמש להצבעה על אובייקט אחד בלבד. אם נדרש יותר מאחד, יש להגדיר מצביע נוסף מאשר להצביע על אובייקט אחר עם המצביע המקורי.    15. יש להשתמש ב-**nullptr** במקום **NULL**.    16. בסיום שימוש במצביע יש להציב לו ערך **nullptr**.    17. יש למזער שימוש במצביעים, עדיף להשתמש בייחוס **&**.    18. עדיף להגדיר מצביע כקבוע כך: **\*const int ptr**.    19. יש להוסיף סופיות לערכים מספריים כדי לסמן **float** וכו'. ל-**double** אין סופית.    20. אין להשתמש בקידומות של טיפוסי משתנים כגון **g\_b** למשתנה מסוג **bool** גלובאלי. מותר רק במה שקשור לקריאת קובץ הקונפיגורציה עקב מגבלות טכניות.    21. גודל וטווח משתנים הוא לפי התקן של **MSVS2015**. ישנו שימוש ב-**sizeof** כדי לבדוק זאת מיד בתחילת התוכנית.    22. יש להשתמש בתבנית **sub\_range** כדי להגביל ערכי משתנים.    23. אם רוצים להשתמש באופרטורים **++** או **–** כחלק מביטוי, אז חובה לפני, כלומר **++x** ולא **x++**. אם זה בשורה נפרדת אז אין חשיבות אם זה **++x** או **x++**.    24. יש לצמצם את השימוש במשתנים גלובליים.    25. משתנה שמשמש כערך חזרה מפונקציה, יקרא **error\_code**, יוגדר ב-**scope** הקטן ביותר האפשרי ובשורה נפרדת ויאותחל לערך חזרה לא חוקי מהפונקציה.    26. יש להשתמש בטיפוס **char**. לא ב-**signed char** או **unsigned char**. כמובן שעדיף **wchar\_t** לתמיכה ביוניקוד.    27. כל חישוב קבוע יש לחשב מראש ולשים את החישוב רק בהערה לדוגמא **screen\_size = {2000}; // 80\*25**. 9. **מחלקות (Classes):**    1. יש לכתוב פונקציית בנאי (**constructor**) לכל מחלקה, גם אם היא ריקה. פונקציה זו תאתחל את כל משתני המחלקה וכל דבר הנדרש כדי לגרום למחלקה להיות מוכן לפעולה אם אפשר.    2. אין צורך לכתוב פונקציית הריסה (**destructor**) לכל מחלקה.    3. עדיף להגדיר מחלקות סטטית ולא דינמית.    4. מחלקות יוגדרו לפי סדר השימוש בהם.    5. פונקציות הבנאים יכתבו לפי סדר השימוש בהם, תוך שימת לב לשימוש הדדי.    6. יש להשתדל להגדיר כל מה שאפשר כ**-protected** ורק את הבנאי ואת מינימום הפונקציות הנחוצות כ-**public**.    7. פונקציות יופיעו ראשונות ורק לאחר מכן המשתנים.    8. כל פונקציה או משתנה צריכים להיות שייכים למחלקה, במידת האפשר. 10. **סדר הרצה:**     1. בדיקה סטטית של תנאי מינימום מבחינת חומרה, תוכנה ומערכת הפעלה.     2. בדיקה דינמית של תנאים נוספים, כגון עותק בודד רץ בזיכרון.     3. ביצוע אתחול לכל מה שדורש אתחול.     4. קריאת כל הקבצים הרלוונטיים ובדיקת חוקיותם. |